Design Environment for Low-Amplitude ThermoAcoustic Engines

DeltaE

Version 3.6

Tutorial and User's Guide

by
Bill Ward and Greg Swift
Los Alamos National Laboratory

LA-CC-93-8
February, 1996
Revision: 6/16/97

Contents

List of Figures

Chapter I

Introduction

DeltaE—Design Environment for Low-Amplitude ThermoAcoustic Engines—is a computer program that can predict how a given thermoacoustic apparatus will perform, or can allow the user to design an apparatus to achieve desired performance. It is currently running on IBM-compatible or Macintosh PCs, VAX minicomputers, and several types of UNIX workstations. It is substantially menu-oriented. Input data can be modified or entered via DeltaE's menu or using any text editor. Results can be examined via the menus, the operating system's text utilities, or any spreadsheet or graphics software.

For good portability, DeltaE is written in FORTRAN-77. The current executable code for IBM-compatibles requires at least a '386 processor, because it uses a DOS extender to create a flat 32-bit memory environment. (An older, version 2.1 DeltaE is still available which requires 333 kbytes of free RAM, and runs comfortably quickly on a 286 with math coprocessor, or anything more sophisticated.) All calculations are performed in double precision.

A How It Works

DeltaE solves the one-dimensional wave equation based on the usual low-amplitude, 'acoustic' approximation. It solves the wave equation in a gas or liquid, in a geometry given by the user as a sequence of segments, such as ducts, compliances, transducers, and thermoacoustic stacks or regenerators. A glance through the figures below will orient the reader to the range of cases that DeltaE can handle.

A solution to the appropriate 1-d wave equation is found for each segment, with pressures and volumetric velocities matched at the junctions between segments. In stacks, the

Figure I.1: Driven, lossy plane-wave resonator.

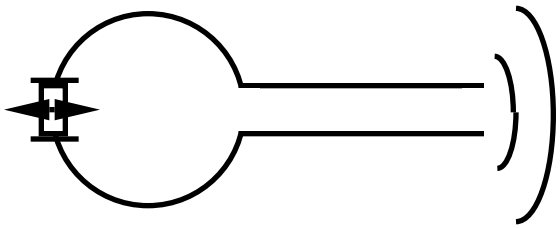


Figure I.2: Driven, radiating Helmholtz resonator.

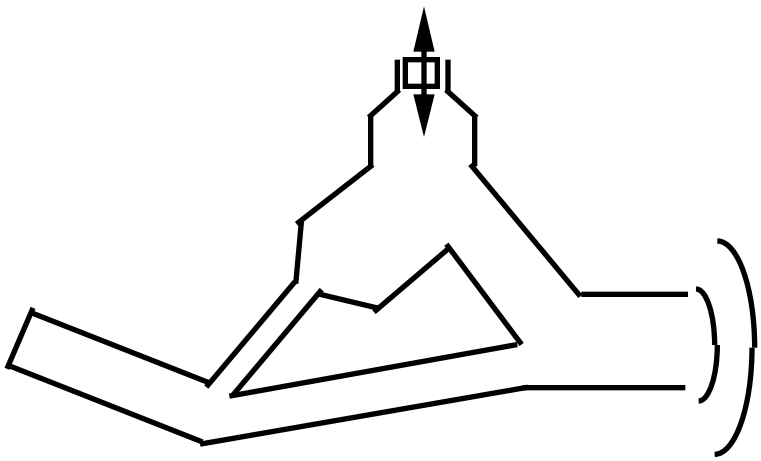


Figure I.3: Duct network.

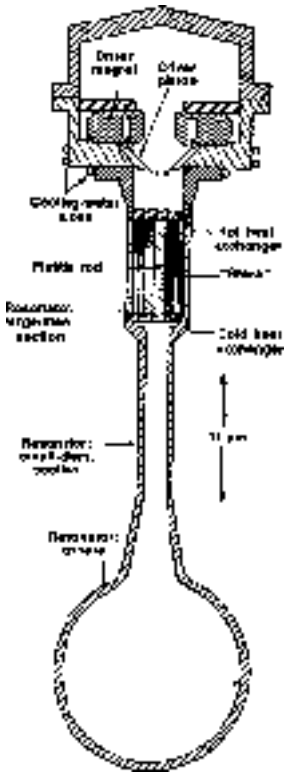


Figure I.4: Thermoacoustic refrigerator (Hofler style). (A=ambient; C=cold)

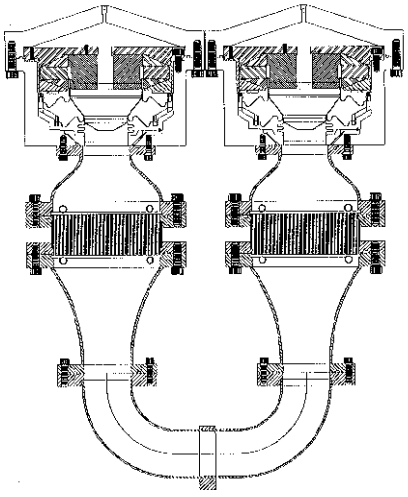


Figure I.5: Thermoacoustic refrigerator (TALSR style)

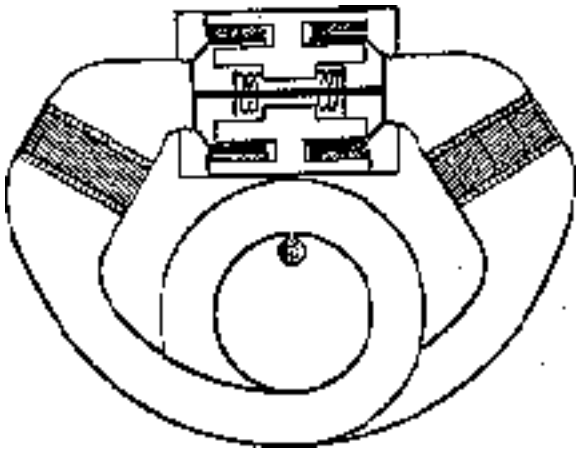


Figure I.6: Thermoacoustic refrigerator (Garrett and Hofler style)

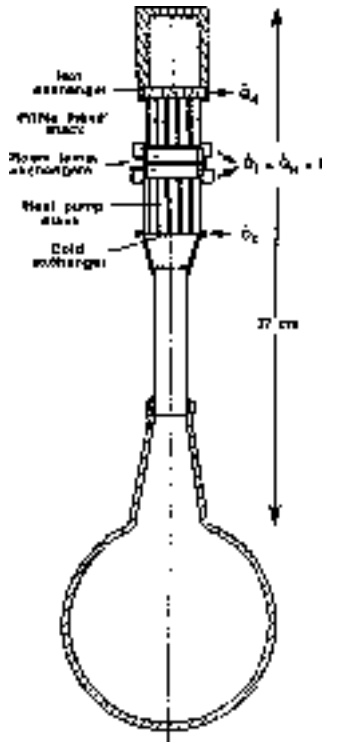


Figure I.7: Beer cooler.

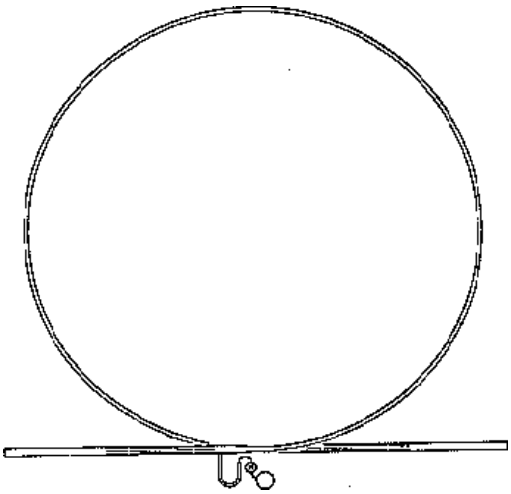


Figure I.8: Thermoacoustically driven orifice pulse-tube refrigerator.

wave-equation solution is found simultaneously with that of the energy-flow equation in order to find the temperature profile as well as the acoustic pressure. The energy flow through stacks is determined by temperatures and/or heat flows at adjacent heat exchangers.

The user of DeltaE enjoys considerable freedom in choosing which variables are computed as 'solutions.' For example, in a simple plane-wave resonator (the first example below), DeltaE can compute the input impedance as a function of frequency, or the resonance frequency for a given geometry and gas, or the length required to give a desired resonance frequency, or even the concentration in a binary gas mixture required to give a desired resonance frequency in a given geometry. Typically, a three to five dimensional solution vector is computed for reasonably complicated thermoacoustic engines, where heat-exchanger temperatures, heat and acoustic powers, efficiencies, etc. are typical solution elements of interest.

Generally, DeltaE does not include any nonlinear effects that arise at high amplitudes, so be cautious using it when Mach numbers or Reynolds numbers are too high. THe principal exception to this rule is the optional turbulence algorithm in ducts, discussed in Chapter V. There are a number of other approximations used, which will be discussed below as we encounter them, and in more detail in Chapter VI.

B How This Manual is Organized

We will teach the use of DeltaE by increasingly complicated examples in Chapters II–IV. Beginning users should read through Chapters II and III until they reach an example

of complexity appropriate for their own case, and then systematically modify one of our examples to suit. Chapter II is just acoustics, without thermoacoustics. It serves to introduce DELTAE's input/output formats and editing and plotting features. Chapter III gives the most complete discussion of the overall principles behind the thermoacoustics computations, and the simplest thermoacoustic examples. The agreement of these examples with published experimental data serves as validation of the code. In Chapter IV, features of DELTAE which are useful in modeling Stirling cycle and more general heat engines are introduced. Chapter V gives the most in-depth discussion of the advanced options of DELTAE. Chapter VI is a segment-by-segment reference chapter for the experienced user, summarizing assumptions built into the computations for each segment, the data format for each segment, and thermophysical properties.

Some of these examples were run on an MS-DOS machine, others on a Mac. (While the menu interface differs, the file formats and displays for both platforms are the same. When there are differences, they will be obvious.) The code was still being debugged and improved while these examples were being run, so there may be some minor errors and formatting oddities in these examples.

We assume that the reader of this manual is very comfortable with linear acoustics and reasonably familiar with thermoacoustics. We will use variables as defined, for example, in the list of symbols in "Thermoacoustic Engines," J. Acoust. Soc. Am. **84**, 1178 (1988).

Chapter II

Basic Acoustics

In this Chapter we use the simplest acoustic segments, such as ducts and endcaps, to introduce DeltaE's basic features.

A Plane-Wave Resonator

We begin with a lossy plane-wave resonator, driven from one end by a piston with a fixed volumetric velocity. We call the input file `planewav.in` (included in the `examples` directory or folder). This input file could have been created from scratch using any text editor, though this one was made by editing one of DeltaE's own output files. (N.B.: A DeltaE input files must always be a plain text file, in the native text format of the machine it is running on. On some systems, integer numbers must be followed by a decimal point, as in the example below. Also, some systems require the last line in the file to be followed by an end-of-line, before the end-of-file occurs.)

```
TITLE     Example 1:  Plane-wave resonator

BEGIN      Initialize things   0
1.000E+05 a Mean P    Pa
100.      b Freq.     Hz
300.      c T-beg      K
1000.     d |p|@0     Pa
90.       e Ph(p)0    deg
1.000E-02 f |U|@0    m^3/s
000      g Ph(U)0    deg
helium

ENDCAP     First end            1
1.000E-02 a Area     m^2
```

```
helium

ISODUCT    Duct                 2
1.000E-02 a Area     m^2
0.354     b Perim     m
5.00      c Length    m
helium

ENDCAP     Second End           3
1.000E-02 a Area     m^2
helium

HARDEND                         4
000     a R(1/Z)
000     b I(1/Z)
helium

! The restart information below was generated by a previous run
! You may wish to delete this information before starting a run
INVARS      2  0  4  0  5
TARGS       2  4  1  4  2
SPECIALS    0
```

Several features of DeltaE input files are illustrated here. Input files should be named something `.IN`. These files consist of a set of segments whose order and format are important. The initial (or 'zeroth') segment is always the `BEGIN` segment, and the last segment is usually `HARDEnd` (or `SOFTEnd` to be discussed in Chapter VI). Intervening segments describe the geometry and other properties of the acoustic engine. The number and order of data in each segment is crucial. All units are MKS. Within each line, only the first number (e.g., "`1.e5`" or "`100.`") or word (e.g., "`helium`" or "`BEGIN`") is important; the rest of the line can be used as a comment field, with, for example, the units or name of the variable whose value appears. Whole-line comments can appear anywhere if they begin with "!" or with 20 or more blanks. Numbers can be in fixed or exponential format. Segment names are all uppercase, and only the first five characters are interpreted (hence, the convention here is to write segment names longer then 5 characters with trailing lower case letters, e.g., `HARDEnd`). All other words must have correct CASE and spelling.

The file shown below works just as well in the computer as the one shown above. However, with fewer comment annotations and only the minimal 5-character segment names, it is harder for humans to follow; it also lacks restart information, so DeltaE might have to prompt us for some more information before proceeding.

```
TITLE

BEGIN
1.e5
100.
300.
1000.
```

```
90.
1.0e-2
0
helium

ENDCA
0.01
helium

ISODU
0.01
354
5.00
helium

ENDCA
0.01
helium

HARDE
0
0.
helium
```

BEGIN sets the stage, in this case, with 1-bar room temperature helium gas being driven at 100 Hz with a pressure amplitude of 1000 Pa and a volume velocity amplitude of 0.01 m^3/s, 90^o out of phase with the pressure.

Since BEGIN has no geometrical properties, an ENDCAp comes next to account for oscillatory thermal losses at the first end of the resonator. An endcap is just a surface area giving dissipation. In this example, because we are beginning with a nonzero volume velocity, ENDCAp can be imagined as the face of the piston.

A lossy isothermal duct ISODUct comes next. Here, we have made the perimeter $\sqrt{4\pi \times \text{area}}$, to make this a circular duct.

The resonator ends with another ENDCAp for oscillatory pressure dissipation.

The input file then ends with the required HARDEnd segment. Its two lines are the real and imaginary parts of the inverse of the end impedance. Since we have set these two equal to zero, this is simply a hard end, with zero volume velocity.

Note that the special segments BEGIN and HARDEnd have no geometrical properties; so ENDCAps are needed to put the thermal dissipation loss at the ends of the resonator.

Figures II.1 show the acoustician's usual cartoon of a driven plane-wave resonator and a pictorial representation of how we modeled this resonator for DeltaE. Throughout this

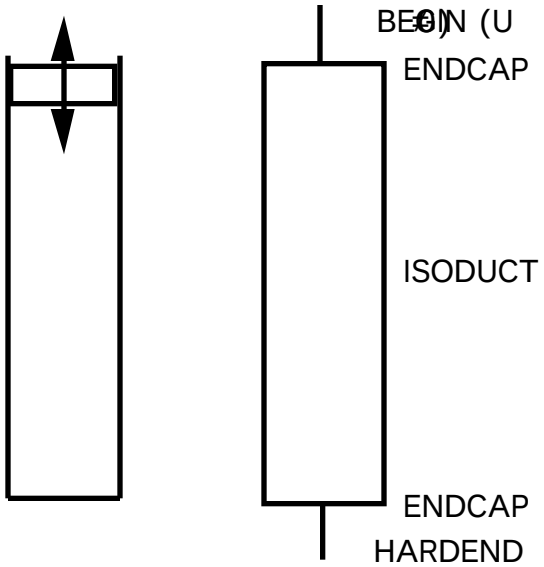


Figure II.1: A plane-wave resonator; conventional and DeltaE representation.

tutorial we use generally conventional symbols to represent ordinary segments such as ducts, horns, and heat exchangers.

This input file overdetermines the acoustic system because only some of the variables listed can be specified independently. You can choose which of these variables DeltaE will regard as fixed, which it will regard as initial guesses at solution values, and, occasionally, which it will ignore.

Execute DeltaE and respond `planwave` to the prompt for an input file (on a MacIntosh, double-click `planwave.in`, or open it using the "New Model" menu). You may also type `deltae planwave`, or `deltae planwave.in`, to load the file directly. After your required "carriage return to continue," DeltaE will respond with:

```
Loading planwave.in . . .
Example 1:  Plane-wave resonator
Ready.
```

DeltaE can accept either `.in` or `.out` files as input files. If you do not type the file suffix, DeltaE looks first for a `.out` file. If it does not find it, it looks next for a `.in` file; if this is not found, DeltaE reprompts for the file name. (On a MacIntosh, all of these steps are replaced by a standard file selection dialog.)

On a keyboard menu system (e.g., PC-compatible, VMS, Unix, etc.), the main menu is displayed next, giving the following options:

```
Main Menu:
   r  (r)un model                      p      (p)lot another parameter
   w  (w)rite current model state      P      (P)lot status summary
   n  (n)ew model input file           c      (c)lear from vectors and plots
   R  (R)estore vectors                C      (C)lear|set all guesses&targets
   E  (E)xtras                         u      (u)se in guess/target vector
   d  (d)isplay                        v      (v)ector status summary
   o  (o)utput to printer              m      (m)odify parameter value
   f  (f)orm feed printer              s      (s)pecial modes editing
   t  (t)hermophysical properties      D      (D)OS command shell
   e  (e)xit DeltaE                    ?      show this menu

MAIN:  (rpwPncRCEudvomfstDe?)>
```

(On the MacIntosh, similar options are available on the menu bar for mouse selection.)

Now select "vector status summary" by typing "`v`":

```
Iteration Vectors Summary:
GUESS      0d          0e
name  BEGIN:|p|@0 BEGIN:Ph(p)
value    1.00E+03     90.
units     Pa          deg

TARGET      4a          4b
name  HARDE:R(1/Z HARDE:I(1/Z
units
value     .00         .00

Potential TARGETS still available are:
Addr Seg:Par-Type    Current Value
```

The `GUESS` vector, which has two components in this example, shows what DeltaE will regard as solution variables: the magnitude and phase of the beginning pressure. Their initial guesses are also shown. (This particular two-dimensional guess vector came from the computer-generated table at the bottom of the input file. This table is explained in Section III.B. If your input file has no such table, DeltaE can make a reasonable guess at the guess variables you might want when you select `(C)lear|set all guesses&targets` with no guesses defined yet.)

Basically, DeltaE integrates the wave equation from `BEGIN` to `END`. We insist that DeltaE refine the two-dimensional `GUESS` vector to find a solution to this acoustics problem by arriving at the `HARDEnd` with zero complex volume velocity. This is accomplished by getting the '0' values of the real and imaginary parts of the inverse of the impedance in the `HARDEnd` segment into DeltaE's two-dimensional `TARGET` vector, as shown in this vector summary table.

The last two lines indicate unselected, still-available targets. These are the only remaining output values for which DELTAE has a reserved input variable available to compare with it. In this model, all such outputs are already in use.

Most of the thought required to successfully run DELTAE occurs while staring at this vector status summary table, trying to figure out which of the variables are appropriate guesses and targets. While the choice of variables is almost entirely arbitrary, as long as the number of guesses equals the number of targets, some choices for the guess vector would have little or no effect on the desired result. For example, allowing DELTAE to try to achieve resonance in a given length by varying the areas of the endcaps would be futile. For the examples in subsequent Chapters, the choice of good guess and target vector members is not always as obvious as it is here.

For now, we will keep these vectors. "Run" the code (type '`r`'), and "`(e)xit`" from DELTAE to the operating system to inspect its results, which DELTAE has put in two new files, `planewav.dat` and `planewav.out`. `planewav.dat` looks like this:

```
 -= Example 1:  Plane-wave resonator                                    =-
 frequency=     100.000Hz     mean pressure=   1.000E+05Pa

  T(K)          p(Pa)               U(m^3/s)             hdot(W)   wdot(W)
   300.0      2933.    2586.6      0.01000  0.00000          14.67     14.67
!----------------------------------  1 ----------------------------------
 ENDCAP    First end
 Heat extracted:   7.333E-02 Watts
   300.0      2933.    2586.6      0.00997 -0.00002          14.59     14.59
!----------------------------------  2 ----------------------------------
 ISODUCT   Duct
 Duct wavvec =(   0.623    ,  -6.207E-03) m^-1
 Heat extracted:    14.5     Watts
   300.0     -2931.   -2588.9     -0.00003 -0.00002           0.07      0.07
!----------------------------------  3 ----------------------------------
 ENDCAP    Second End
 Heat extracted:   7.331E-02 Watts
   300.0     -2931.   -2588.9      0.00000  0.00000           0.00      0.00
!----------------------------------  4 ----------------------------------
 HARDEND   Final
 inverse impedance (rho a U/p A)=(  -2.485E-08,   2.983E-08)

   300.0     -2931.   -2588.9      0.00000  0.00000           0.00      0.00
```

Examination of `planewav.dat` will show that the solution is $\Re(p) = 2933$ Pa, $\Im(p) = 2587$ Pa. Also shown are temperature, complex p_1, and complex U_1, work flow, and energy flow at the beginning and end of each segment. You can see, for instance, that the driver delivers 14.7 Watts of power to the resonator, that 0.07 Watts is absorbed on each end, and that 14.5 Watts is absorbed by the duct. The work absorbed in isothermal segments such as these is extracted as heat, *e.g.* by a water jacket in the real world.

The output model file, `planewav.out`, is shown below:

```
TITLE     Example 1:  Plane-wave resonator
!--------------------------------  0 --------------------------------
 BEGIN      Initial
  1.0000E+05 a Mean P    Pa              3911.     A |p|@0  G( 0d)     P
   100.0     b Freq.     Hz              41.41     B Ph(p)0 G( 0e)     P
   300.0     c T-beg     K
   3911.     d |p|@0     Pa     G
   41.41     e Ph(p)0    deg    G
  1.0000E-02 f |U|@0    m^3/s
  0.0000     g Ph(U)0    deg
 helium     Gas type
 ideal      Solid type
!--------------------------------  1 --------------------------------
 ENDCAP     First end
  1.0000E-02 a Area      m^2              3911.     A |p|       Pa
                                          41.41     B Ph(p)     deg
                                       9.9719E-03 C |U|     m^3/s
                                         -0.1425     D Ph(U)     deg
                                          14.59     E Hdot      W
 helium     Gas type                      14.59     F Work      W
 ideal      Solid type                 -7.3327E-02 G HeatIn    W
!--------------------------------  2 --------------------------------
 ISODUCT    Duct
  1.0000E-02 a Area      m^2              3910.     A |p|       Pa
  0.3540     b Perim      m              -138.5     B Ph(p)     deg
   5.000     c Length     m             3.7491E-05 C |U|     m^3/s
                                         -138.5     D Ph(U)     deg
                                       7.3300E-02 E Hdot      W
 helium     Gas type                   7.3300E-02 F Work      W
 ideal      Solid type                   -14.52     G HeatIn    W
!--------------------------------  3 --------------------------------
 ENDCAP     Second End
  1.0000E-02 a Area      m^2              3910.     A |p|       Pa
                                         -138.5     B Ph(p)     deg
                                       9.2828E-09 C |U|     m^3/s
                                         -8.743     D Ph(U)     deg
                                      -1.1617E-05 E Hdot      W
 helium     Gas type                  -1.1617E-05 F Work      W
 ideal      Solid type                -7.3312E-02 G HeatIn    W
!--------------------------------  4 --------------------------------
 HARDEND    Final
  0.0000     a R(1/Z)          = 4G?     3910.     A |p|       Pa
  0.0000     b I(1/Z)          = 4H?    -138.5     B Ph(p)     deg
                                       9.2828E-09 C |U|     m^3/s
                                         -8.743     D Ph(U)     deg
                                      -1.1617E-05 E Hdot      W
                                      -1.1617E-05 F Work      W
                                      -2.4853E-08 G R(1/Z)
 helium     Gas type                   2.9829E-08 H I(1/Z)
 ideal      Solid type                   300.0     I   T       K

! The restart information below was generated by a previous run
! You may wish to delete this information before starting a run
! where you will (interactively) specify a different iteration
! mode.  Edit this table only if you really know your model!
```

```
INVARS      2  0  4  0  5
TARGS       2  4  1  4  2
SPECIALS    0
```

(Some digits of lesser significance in DeltaE output examples in this manual may vary from the numbers which you get running your version of the code. This primarily due to differences in floating point arithmetic hardware and software between different compilers and computers, and the finite tolerance against which DeltaE measures the accuracy of its results. This can be reduced from the default (see Chapters 5 and 6 for details) to force iterations to continue until a greater degree of precision is achieved. For lower tolerance levels, all significant digits in the finished output file will agree for all versions of the code with relatively straightforward models.)

Examination of `planewav.out` will show that it is a slightly modified version of our `planewav.in`: It includes the solution values for magnitude and phase of beginning pressure (3911 Pa. 41.41^o), replacing our original guesses (1000 Pa. 90^o). DeltaE would have made this file as it is even if we had used the bare-bones, unannotated version of the input file. In `*.out` files, DeltaE numbers the segments and 'letters' the lines in each segment for our convenience, and displays names and units for all variables. It adds the obscure table at the end that reflects our choice of guess and target vectors. The format of DeltaE's `.out` file is acceptable as an input file, so using it as such can save the user a lot of work.

As a new example, we will find the resonance frequency f, which we guess is near 100 Hz. We'll use the same old `planewav.in`, so execute DeltaE again and select it. Display the vector status summary again.

```
Iteration Vectors Summary:
GUESS      0d          0e
name  BEGIN:|p|@0 BEGIN:Ph(p)
value    1.00E+03     90.
units     Pa          deg
TARGET      4a          4b
name  HARDEND  :R(1/Z HARDEND  :I(1/Z
units
value     .00         .00
```

Now we want f, $|p|$ in segment `BEGIN` as the 2 components of the guess vector. We will fix the phase of the beginning p_1 at 0, because having p_1 and U_1 in phase at the driver is the condition for resonance. So we want to change this table to look like this:

```
Iteration Vectors Summary:
GUESS      0b          0d
name  BEGIN:Freq. BEGIN:|p|@0
value    1.00E+02    1.00E+03
units    Hz        Pa
```

```
TARGET      4a          4b
name  HARDEND  :R(1/Z HARDEND  :I(1/Z
units
value     .00         .00
```

We can make this change in guess vector by a "(c)lear" of 0e from the guess vector; "(u)se" 0b instead; and "(m)odify" 0e to be zero degrees instead of 90^o. (These vectors happen to be identical to DeltaE's default, so we could have generated this table by selecting (C)lear|set twice—first to wipe out the old vectors, and then again to set the defaults.) Now, "(r)un" the calculation. Inspect the results by using "(d)isplay" from within DeltaE (or by escaping to the operating system, as you did before). You will find that the resonance frequency is 100.9 Hz.

If you can't remember the number-letter code for the variable you want when modifying the vectors, "(d)isplay" all segments, or "(d)isplay" a selected segment number to see a list of the variables. For example, "(d)isplay" segment 0 to find out which number-letter code is for frequency:

```
INPUT     # ParType Units  Status      OUTPUT    # ParType Units  Status
--------------------------------  0 --------------------------------
BEGIN      Initialize things
1.000E+05 a Mean P    Pa               .000     A |p|@0     0d      P
100.      b Freq.     Hz               .000     B Ph(p)0    0e      P
300.      c T-beg     K
1.000E+03 d |p|@0     Pa      G
90.0      e Ph(p)0    deg     G
1.000E-02 f |U|@0    m^3/s
000      g Ph(U)0    deg
helium     Gas type
ideal      Solid type
```

If you incorrectly remember a number-letter code and are stuck in a modification, you can either type "return" repeatedly to accept existing values, or type "x" to escape. (If you're really stuck, control-C will escape from nearly anywhere.)

DeltaE can use any physically appropriate variables in the guess vector. You can determine what temperature makes the model resonate at 100 Hz, by putting 0c in the guess vector. (The answer is 290.7 Kelvin.) Or, by putting 2c in the guess vector, we could have found out what length the model needs to be to resonant at 100 Hz at 300 K. An advanced feature to be discussed in Chapter V allows use of the concentration in a binary gas mixture to be used (as a member of the guess vector) so that we could determine the argon concentration that would be required in the helium to make the resonance at 100 Hz.

B Plotting

DeltaE allows for plotting by automatically incrementing (or decrementing) one or two independent variables, and tabulating these together with one or more output variables in a file named something`.plt`. Users can then manipulate and/or plot that file with their favorite graphics or spreadsheet software. We illustrate these features with a continuation of the same example, a plane-wave resonator.

We use the same input file as before, `planewav.in`. Execute DeltaE and choose this as input file. Check the vector status summary:

```
Iteration Vectors Summary:
 GUESS      0d          0e
 name  BEGIN:|p|@0 BEGIN:Ph(p)
 value    1.00E+03     90.
 units     Pa          deg
TARGET      4a          4b
 name  HARDEND  :R(1/Z HARDEND  :I(1/Z
 units
 value     .00         .00
```

Now inspect the Plotted parameter summary (type capital "P"):

```
Dependent Variables (outputs):
PLOTS     0A        0B
 name  BEGIN:|p|@0 BEGIN:Ph(p)
 units    Pa        deg
```

Keep these parameters as the dependent variables to be plotted (they are copies of the guesses). To set up the independent variables, select "plot another variable." We will make a two-dimensional plot, letting f go from 80 Hz to 339.5 Hz in 1.5-Hz steps in the inner loop, and using two values of mean pressure—1000 Pa and 100,000 Pa—in the outer loop. DeltaE prompts for these entries in the "range" selection. As before, if you can't remember the segment-number and line-letter codes for frequency and mean pressure, "(d)isplay" segment 0 to find out. After entering these values, check the Plotted parameter summary again:

```
Dependent Variables (outputs):
PLOTS     0A        0B
 name  BEGIN:|p|@0 BEGIN:Ph(p)
 units    Pa        deg
Indpendent Variables (inputs):
 Outer loop:   0a  BEGIN:Mean  Beg=  1.00E+03 End=  1.00E+05 Step=  9.90E+04
 Inner Loop:   0b  BEGIN:Freq. Beg=   80.     End=  3.40E+02 Step=   1.5
```

Now do a `(r)un`. DELTAE will step through the variables selected (taking a minute or two on a 286). When it has finished, exit to the operating system, and find two new files. The file `planewav.des` gives headings of what has been tabulated:

```
BEGIN:Mean  BEGIN:Freq. BEGIN:|p|@0 BEGIN:Ph(p)
     Pa         Hz         Pa         deg
     0a         0b         0A         0B
```

and `planewav.plt` is the table of values:

```
*  1000.0       80.00      1000.0       90.00
*  1000.0       81.50      1000.0       90.00
*  1000.0       83.00      1000.0       90.00
*  1000.0       84.50      1000.0       90.00
*  1000.0       86.00      1000.0       90.00
   1000.0       87.50       5.095       23.89
   1000.0       89.00       5.398       16.33
   1000.0       90.50       5.598       8.033
   1000.0       92.00       5.656     -0.6461
     ⋮

   1.0000E+05   87.50      -370.7       265.4
   1.0000E+05   86.00      -328.6       265.8
   1.0000E+05   84.50      -293.7       266.0
   1.0000E+05   83.00      -264.1       266.3
   1.0000E+05   81.50      -238.6       266.4
   1.0000E+05   80.00      -216.4       266.6
```

Notice that the first few lines of `planewav.plt` begin with "$*$" and their values do not make much sense. The asterisk signifies that DELTAE did not converge to a solution it judged accurate, so it suspects that these data may be invalid. (We could have avoided this by giving DELTAE a better initial guess for $|p_1|$ and phase(p_1), but we chose instead to let DELTAE seek a solution.) Warning messages appeared on the screen as each of these points was calculated. Once it converged (at 87.5 Hz) DELTAE was in no danger of getting lost again because it always uses its previous solution as its initial guess, and with such small frequency increments the previous solution is an excellent guess. Notice also, that DELTAE alternates the order in which it calculates the points of the inner loop (frequency, here). This process is motivated by the quality of initial guesses; 'zig-zagging' thus, DELTAE must spend only a brief time searching for the start point of the inner loop each time it begins a new cycle.

We brought this file into a spreadsheet/graphics program to fix it up for plotting. We removed the asterisked lines; we also removed minus signs from $|p_1|$whenever they occurred, adding 180^o to phase(p_1) in those cases to improve the looks of the the graphs. We also plotted $|p_1|/p_m$ (instead of just $|p_1|$). The resulting plots are shown in Figs. II.2. (The lower quality-factor curves are for the lower mean pressure, of course.)

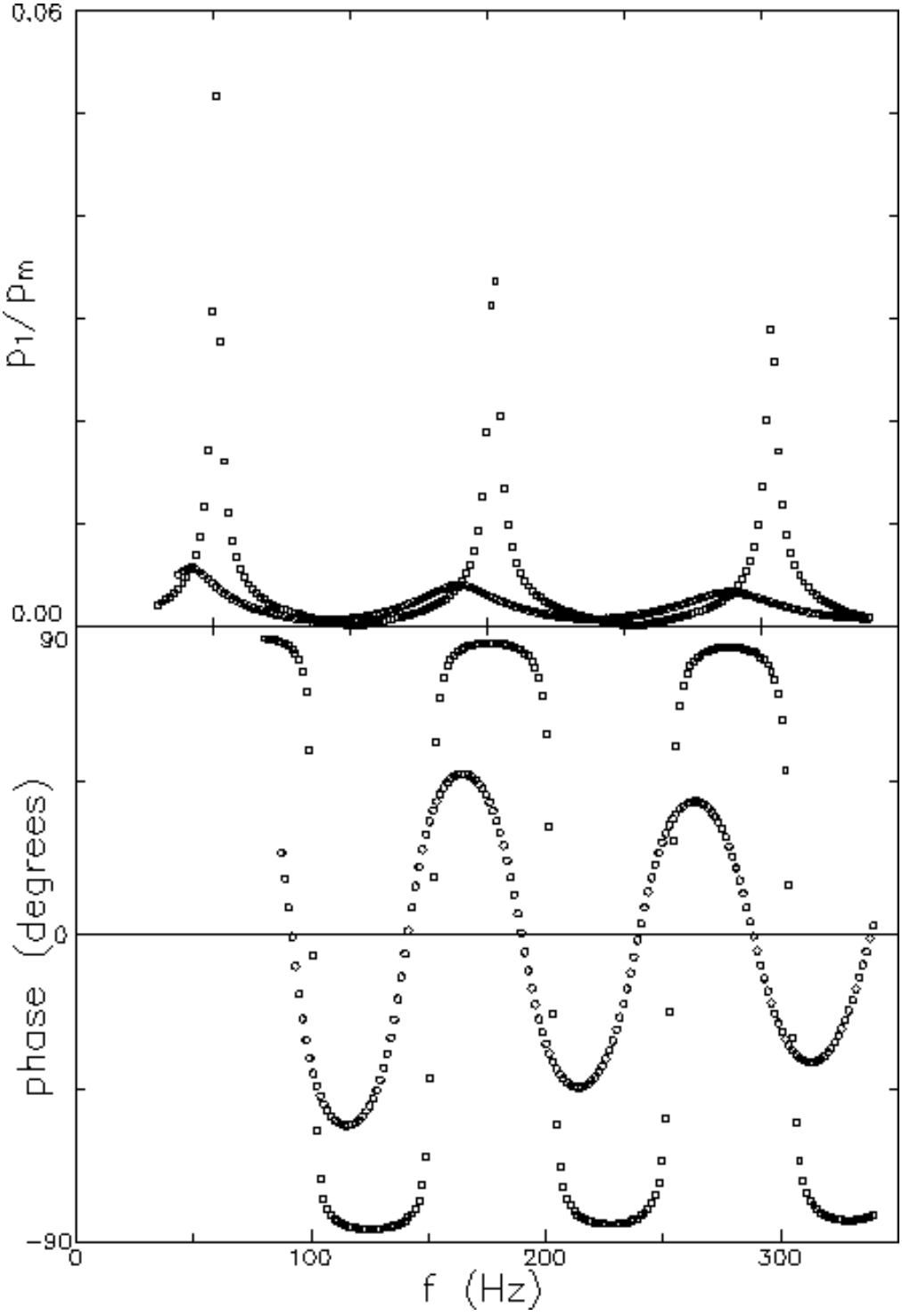


Figure II.2: Pressure and phase vs frequency for the plane-wave resonator.

C Further Simple Features

Here we describe some additional DELTAE features which are relevant to purely acoustic (not thermoacoustic) apparatus. A list of the most commonly used purely acoustic segment types (including those introduced previously) is given below. More detailed descriptions of each are given in Chapter VI.

`TITLE` Required at the top; comment field is retained in all `.DAT` and `.OUT` files.

`BEGIN` Required immediately after `TITLE`. This is the "zeroth" segment. It defines global parameters such as mean pressure and frequency, and initial conditions for p_1 and U_1.

`ENDCAp` A surface area with oscillatory-pressure loss in its thermal penetration depth. Usually used at ends of ducts.

`ISODUct` An isothermal duct, with losses. Separate entry of area and perimeter accommodates ducts of any cross-sectional shape.

`ISOCOne` A cone to adapt between ducts of different sizes. Uses lossy Webster horn equation.

`COMPLiance` A compliance. With oscillatory-pressure losses on surface.

`IMPEDance` A lumped-parameter series impedance.

`IDUCEr` and `VDUCEr` Current-driven and voltage-driven transducers, with parameters independent of frequency.

`ISPEAker` and `VSPEAker` Current-driven and voltage-driven electrodynamic transducers, parameterized by mass, B-L product, etc., so that impedance coefficients depend on frequency.

`IEDUCer` and `VEDUCer`, `IESPEaker` and `VESPEaker` The four transducers above are in *branched* configurations, where pressure is unchanged by the transducer. These `E`nclosed versions are their *series* counterparts, where volumetric velocity remains constant across the segment.

`BRANCh` A frequency-independent side-branch impedance.

`OPNBRanch` A frequency-dependent side-branch impedance with the characteristic radiation impedance of a duct opening into an infinite or semi-infinite space.

`HARDEnd` One of the allowed final segments. Used to set $U_1 = 0$ through use of the inverse of the acoustic impedance in the `TARGET` vector.

`SOFTEND` The other allowed final segment. Used to set $p_1 = 0$ through use of the acoustic impedance in the `TARGET` vector. Very useful for defining mirror-image planes in symmetric apparatus with pressure nodes at center of symmetry.

Each segment must have a gas type and a solid type (even segments that don't actually use such information, such as `BRANCh`). At present, DeltaE supports air, helium, neon, He-Xe, He-Ar, and He-Ne mixtures (see Chapter VI), hydrogen, deuterium, nitrogen, carbon dioxide, natural-gas combustion products (i.e., chimney exhaust), liquid sodium, and eutectic liquid NaK as gases. Solids include Kapton, mylar, stainless steel, molybdenum, tungsten, copper, nickel, and ideal. An ideal solid is one that has infinite heat capacity, density, and thermal conductivity. If no solid type is given in the input file, DeltaE will assign the ideal solid type. There is also a mechanism for specifying additional, user-defined fluids and solids; details are given in Chapter VI.

The `sameas nl` feature allows reference to a value (either a number or a gas or solid type) in another segment. This helps prevent typographical errors in the input file, and is especially useful in linking dimensions of adjacent segments which you might want to vary all together while plotting, such as areas of all segments when increasing the size of the apparatus. You can give just the segment number, if the parameter letters are the same (e.g., "`sameas 0`" is often the gas type in all segments after the zeroth segment), or segment number and line letter (e.g., "`sameas 3a`"). If you try to use `sameas` for two different types of variables—making a length the same as an area, for example—DeltaE will give an error message and abort. The following example is for a closed resonator composed of two ducts joined by a cone:

```
TITLE    illustrating use of sameas

BEGIN      Initial              0
  1.380E+06 a Mean P    Pa
   132.     b Freq.     Hz
   586.     c T-beg     K
  6.639E+04 d |p|@0     Pa
   .000     e Ph(p)0    deg
   .000     f |V|@0    m^3/s
   .000     g Ph(V)0    deg
helium     Gas type

ENDCAP     Hot End              1
sameas 2a   a Area     m
sameas  0  Gas type

ISODUCT    Hot Duct             2
1.292e-2    a Area     m
   .403     b Perim    m
1.0         c Length   m
sameas  0  Gas type

ISOCONE     adapter between ducts
```

```
sameas 2a   a AreaI   m^2
sameas 2b   b PerimI   m
   .100     c Length   m
sameas 4a   d AreaF   m^2
sameas 4b   e PerimF   m
sameas 0

ISODUCT    Cold Duct            4
0.323e-2  a Area      m^2
0.2015    b Perim      m
1.0       c Length     m
sameas 0

ENDCAP     Cold End             5
sameas  4a  a Area    m^2
sameas 0

HARDEND                         6
   .000     a R(1/Z)
   .000     b I(1/Z)
sameas 0
```

When you access a parameter specified by `sameas` using `(m)odify`, or `(p)lot` to make it an independent plot variable, or `(u)se` it in a guess vector, the `sameas` relationship is severed and the parameter is given its current actual value. This is required because the value at this point will be changed (either by you, or by DeltaE). But if a variable that is the root of several `sameas` references is caused to change in any of these three ways, all `sameas` references to this root within the model will change with it.

Chapter III

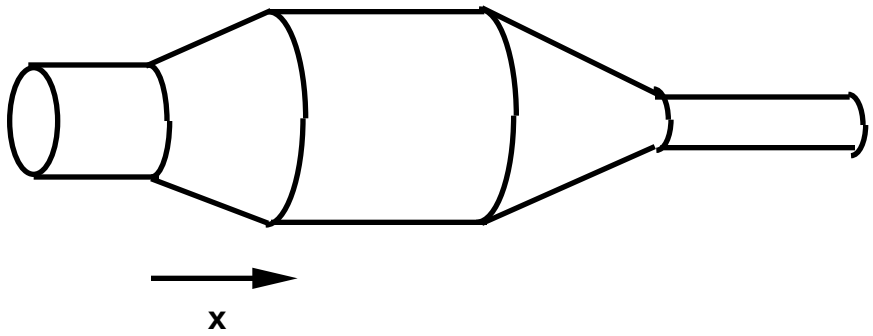
Thermoacoustics

The examples given in the previous Chapter were relatively simple cases of linear acoustics. In this Chapter, we introduce the full thermoacoustic computing power of DeltaE. After discussing the principles and assumptions that are built into the computation, we present example calculations.

A Principles of Computations

DeltaE deals with one-dimensional strings of acoustic and thermoacoustic elements (see Chapter V for branches), so the one-dimensional wave equation is of the greatest importance. We always assume a time dependence of $e^{i\omega t}$, so the wave equation is a second-order differential equation for the complex pressure amplitude $p_1(x)$:

$$p_1 + \frac{a^2}{\omega^2}\frac{d^2 p_1}{dx^2} = 0. \tag{III.1}$$



More complexity is added, as needed, for given geometries. For example, in a duct, the wave equation is

$$(1+\frac{1-i}{2}\frac{\Pi}{A}\frac{\gamma-1}{1+\epsilon_s}\delta_\kappa)p_1+\frac{a^2}{\omega^2}(1-\frac{1-i}{2}\frac{\Pi}{A}\delta_\nu)\frac{d^2p_1}{dx^2}=0, \tag{III.2}$$

where A is the cross-sectional area of the duct, Π is its perimeter, and ϵ_s is a correction for thermal properties of the duct wall that is usually negligible. In a shallow lossy horn, where A and Π themselves depend on x, the wave equation is

$$\left[1+\frac{1-i}{2}\frac{\Pi}{A}\frac{\gamma-1}{1+\epsilon_s}\delta_\kappa\right]p_1+\frac{a^2}{\omega^2}\frac{1}{A}\frac{d}{dx}\left(\left[1-\frac{1-i}{2}\frac{\Pi}{A}\delta_\nu\right]A\frac{dp_1}{dx}\right)=0. \tag{III.3}$$

In a stack, we use Rott's wave equation:

$$(1+\frac{(\gamma-1)f_\kappa}{1+\epsilon_s})p_1+\frac{\rho_m a^2}{\omega^2}\frac{d}{dx}(\frac{1-f_\nu}{\rho_m}\frac{dp_1}{dx})-\beta\frac{a^2}{\omega^2}\frac{(f_\kappa-f_\nu)}{(1-\sigma)(1+\epsilon_s)}\frac{dT_m}{dx}\frac{dp_1}{dx}=0. \tag{III.4}$$

In DELTAE, the computation uses the wave equation that is appropriate for each segment. Within each segment, wave propagation is controlled by local parameters such as area and perimeter. Although DELTAE uses analytic solutions to the wave equation for some of the simplest segment types, it often must integrate the wave equation numerically, so it is generally correct to imagine DELTAE beginning at the `BEGIN` segment and numerically integrating the wave equation through each segment, in turn, to the `HARDEnd` or `SOFTEnd`, using local parameters, such as area and perimeter, as it goes.

It is sometimes easier to think of the second-order wave equation as two coupled first-order equations in pressure p_1 and volume velocity U_1 :

$$\begin{aligned} U_1 &= \frac{iA}{\omega\rho}\frac{dp_1}{dx}; \\ p_1 &= \frac{i\rho a^2}{\omega A}\frac{dU_1}{dx}. \end{aligned} \tag{III.5}$$

From this point of view it is easier to understand DELTAE's use of continuity of p_1 and U_1 to pass from the end of one segment to the beginning of the next. This point of view is taken in the review article, "Thermoacoustic engines and refrigerators" by G.W. Swift, to be published in the *Encyclopedia of Applied Physics.*

Either way, however, it is clear that the solution $p_1(x)$, $U_1(x)$ is only determined uniquely if four real boundary conditions are imposed, because the governing equation is second order in a complex variable. This is true whether considering a single segment or a one-dimensional string of segments with each joined to its neighbor(s) by continuity of p_1 and U_1. If all four boundary conditions are given at one end of the apparatus (i.e., if we know

the complex p_1 and complex U_1 at the `BEGIN` segment) then the integration is utterly straightforward. But usually some of the boundary conditions are given elsewhere. For example, in the plane-wave resonator in the previous Chapter, the boundary conditions were $U_1 = (0.01,\ 0)$ m/s at the `BEGIN` segment, and $U_1 = (0,\ 0)$ at the `HARDEnd`. In such conditions DELTAE uses a shooting method,[1] by guessing any unknowns among the four numbers defining p_1 and U_1 at the `BEGIN` segment, integrating to the `HARDEnd`, comparing the results with the boundary conditions imposed at the `HARDEnd`, and adjusting its guesses until it comes out right.

The boundary conditions imposed at the `HARDEnd` are in DELTAE's `TARGET` vector. The unknown conditions at the `BEGIN`ning, which DELTAE is supposed to find, are in DELTAE's `GUESS` vector. The number of elements in the `TARGET` vector must equal the number of elements in the `GUESS` vector; otherwise the system is over- or under-determined.

One of DELTAE's most powerful features is that the elements of the `GUESS` vector are not limited to the conventional choices consisting of real and imaginary parts of p_1 and U_1 at the `BEGIN`ning. Any variables that have an effect on the `TARGET` vector variables can be used. This enables DELTAE to calculate resonance frequencies, geometrical dimensions, temperatures, or even concentration in binary gas mixtures in order to satisfy given boundary conditions.

To add thermoacoustic computation ability to this linear acoustic picture, only one more equation is required, i.e., that for the temperature $T_m(x)$. As for $p_1(x)$ and $U_1(x)$, the equation for $T_m(x)$ depends on the type of segment, and continuity of $T_m(x)$ is imposed at the junctions between segments. Most segments, such as isothermal ducts and cones, obey simply $dT_m/dx = 0$. Stacks have a more complicated, but still only first-order, differential equation for $T_m(x)$:

$$\begin{aligned}
\dot{H}_2 &= \frac{A_{\text{fluid}}}{2\omega\rho_m}\Im\left[\frac{d\tilde{p}_1}{dx}p_1(1-\tilde{f}_\nu-\frac{T_m\beta(f_\kappa-\tilde{f}_\nu)}{(1+\epsilon_s)(1+\sigma)})\right] \\
&+ \frac{A_{\text{fluid}}c_p}{2\omega^3\rho_m(1-\sigma)}\frac{dT_m}{dx}\frac{dp_1}{dx}\frac{d\tilde{p}_1}{dx}\Im\left[\tilde{f}_\nu+\frac{(f_\kappa-\tilde{f}_\nu)(1+\epsilon_s f_\nu/f_\kappa)}{(1+\epsilon_s)(1+\sigma)}\right] \\
&- (A_{\text{fluid}}K+A_{\text{solid}}K_s)\frac{dT_m}{dx} \qquad \text{(III.6)}
\end{aligned}$$

So, for thermoacoustic calculations, DELTAE integrates from `BEGIN`ning to `HARDEnd`, with respect to five real variables: real $T_m(x)$, complex $p_1(x)$, and complex $U_1(x)$. It uses

[1] Precisely speaking, DELTAE forms a system of nonlinear equations from the model using the targets that the user selects and manipulates the guesses to drive the differences between the targets and results to zero. The routine incorporated in the code is called `DNSQ`, and it is part of the SLATEC Common Mathematical Library, which is freely available through the internet software repository at "http://www.netlib.org". The algorithm used is a modification of the Powell hybrid method.

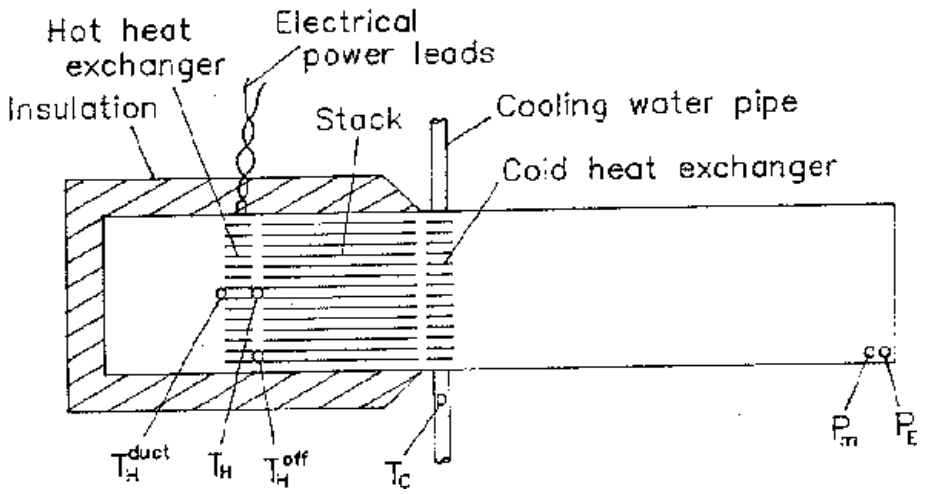


Figure III.1: 5-inch engine.

the appropriate wave equation and temperature equation for each segment. Within each segment, wave propagation is controlled by local parameters, such as area and perimeter, and by global parameters, such as frequency and mean pressure. Spatial evolution of temperature profile is also controlled by such local parameters, which include energy flow. (Energy flow includes both longitudinal conduction in the solid elements of an element, and enthalpy flow. Enthalpy flow is a conceptually difficult parameter because it depends on the heat flows into adjacent heat exchangers and on work flowing along the apparatus. It is therefore like the frequency in a resonant system in that it is a parameter that controls wave propagation in a segment but whose value is determined elsewhere.)

B The 5-Inch Engine

The 5-inch engine is described in detail in *J. Acoust. Soc. Am.* **92**, 1551 (1992). The device described there is used to illustrate the fully thermoacoustic capabilities of DELTAE here; we reproduce some of the figures in that paper.

The apparatus is shown in FIg. III.1. Beginning with the input file (`5inch.in`, in the `examples` directory) to illustrate stack and heat exchanger segment types:

```
TITLE   Five-Inch Thermoacoustic Engine

BEGIN   Initial
13.8e5  (Pa) mean pressure
100.    (Hz) freq
500.    Starting Temp
8.e4    Mag(Pa) acoustic pressure @x=0
0.      Phase (deg) acoustic pressure @x=0
0.      Mag(vdot) vol. veloc @x=0
0.      Phase (deg) vol. veloc @x=0
```

```
helium  gas type

ENDCAP  Hot End
0.01292 (m^2) area
sameas 0  gas type

ISODUCT Hot Duct
sameas 1 (m^2) total area
0.403   (m) perim
0.279   (m) length
sameas 0  gas type

HXFRST Hot HX
sameas 1  (m^2) total area
0.393    gas area/total area
0.060    (m) length
0.483e-3 (m) y0 (this is half the gap)
2210.20  (W) heat
550.     (K) temperature
sameas 0   gas type

STKCIRC Honeycomb Stack
sameas 1  (m^2) total area
0.81     gas area/total area
0.279    (m) length
0.50e-3  (m) radius of each 'circular' pore
0.05e-3  (m) L:half of sht thcknss
sameas 0    gas type
stainless stack material

HXLAST Cold HX
0.01267  (m^2) total area
0.486    gas area/total area
0.0508   (m) length
0.406e-3 (m) y0
0.0      (W) heat
303.     (K) temperature
sameas 0    gas type

ISODUCT Cold Duct
sameas 5  (m^2) total area
0.399    (m) perim
3.654  (m) length
sameas 0    gas type

ENDCAP  Cold End
sameas 5  (m^2) area
sameas 0   gas type

HARDEND
0.      Re(zinv)
0.      Re(zinv)
sameas 0   gas type
```

Of the three types of heat exchanger segments, only two are shown here. `HXFRSt` comes before a stack; `HXLASt` comes after a stack; `HXMIDl` comes between stacks. They differ in whether the heat flow is considered to be an input (possibly a guess) or a result.

All `HX`'s are assumed to have parallel-plate geometry, with plate spacing $2y_o$. Other geometry is given in straightforward format. Wave propagation through heat exchangers is computed using a complex wavevector including both viscous and thermal dissipation in this geometry.

One additional feature of `HX`'s is heat flow. Positive heat flows into the apparatus. In `HXFRSt` and `HXMIDl`, the heat flow determines the change in energy flux in the heat exchanger. Thus, in these cases heat flow can be either be fixed (and optionally, an independent plot variable) or it can be a member of the guess vector . In `HXLASt`, the change in energy determines the heat flow, so the heat can be a result or target (and optionally a dependent plot variable). This example uses the hot heat exchanger's heat flow as the independent plot variable and the cold heat exchanger's heat flow as a simple result that is largely ignored here.

A second additional feature of the `HX`'s is the temperature difference between the mean-gas temperature and the heat exchanger metal temperature, proportional to the heat flow. Its dependence on the geometry of the heat exchanger is given in Chapter VI. [This temperature difference can presently be computed only within an accuracy of about a factor of 2, even in the acoustic approximation; nevertheless, it is included, to prevent naive users from being led to designs with heat exchangers of negligible surface area that have negligible losses and that would appear to have no disadvantages if the temperature difference were not included. Future revisions of DeltaE, hopefully, will have an accurate calculation algorithm for this effect. Meanwhile, however, if you prefer not to use this feature, use the gas mean temperature instead of the metal temperature by using a `FREETarget` (see Section VI.A) to access the temperature in the adjacent stack segment (parameter G or H).] Metal temperature can be a target or a result. In this example, the cold heat exchanger's temperature is used as a target and the hot heat exchanger's temperature is used as a result and plotted.

Seven types of stacks are allowed: `STKSLab` for parallel-plate geometry, `STKCIrc` for circular pores, `STKREct` for rectangular and square pores, `STKPIns` for pin-array stacks, and `STKDUcts` for boundary-layer-approximation stacks (with all dimensions much greater than thermal and viscous penetration depths). `STKSCreen` and `STKPOwerlaw` for regenerators for Stirling systems will be introduced in Chapter IV. The geometry for each type is given as shown in Chapter VI below. Evolution of T_m, p_1, and U_1 are computed as described in: *J. Acoust. Soc. Am.* **84**, 1145 (1988), *J. Acoust. Soc. Am.* **92**, 1551 (1992), *J. Acoust. Soc. Am.* **90**, 3228 (1991), and *J. Acoust. Soc. Am.* **94**, 941 (1993).

You can execute DeltaE using the input file above and use `(C)lear|set` to ask for default targets:

```
No vectors defined...do you want enable a default
set of targets&guesses for this model? (y/n)   y
Is this a prime-mover or a heat pump(p|h)?  p
```

We selected '`p`' because this device is a prime mover. Examining the vector summary, we find:

```
Iteration Vectors Summary:
 GUESS      0b          0c          0d
 name  BEGIN:Freq. BEGIN:T-beg BEGIN:|p|@0
 value    1.00E+02    5.00E+02    8.00E+04
 units     Hz          K           Pa
TARGET      5f          8a          8b
 name  HXLAS:Est-T HARDE:R(1/Z HARDE:I(1/Z
 units      K
 value    3.03E+02     .00         .00

Potential TARGETS still available are:
Addr Seg:Par-Type    Current Value
 3f HXFRST:Est-T =   550.0         K
 5e HXLAST:HeatIn=   .0000         W
```

DeltaE has made good default choices for guess and target vector elements. The default is a three-dimensional search, with end impedance and cold heat-exchanger temperature as targets.

Other choices could be made for this table. For instance, the cold-duct length could be substituted for the frequency in the guess vector. A fourth component, such as the hot heat-exchanger temperature `3f` could be added to the target vector and, simultaneously, the hot heat-exchanger heat `3e` could be added to the guess vector. For now, however, these vectors will be left as they are, since they reflect the point of view adopted in *J. Acoust. Soc. Am.* **92**, 1551 (1992).

If you run this case, you will get the following `.dat` file:

```
 -= Five-Inch Thermoacoustic Engine                                        =-
 frequency=     121.023Hz     mean pressure=   1.380E+06Pa

  T(K)          p(Pa)               U(m^3/s)             hdot(W)   wdot(W)
   557.7     73450.       0.0      0.00000  0.00000          0.00      0.00
!----------------------------------  1 ----------------------------------
 ENDCAP    Hot End
 Heat extracted:    1.22     Watts
   557.7     73450.       0.0     -0.00003  0.00000         -1.22     -1.22
```

```
!----------------------------------  2 ----------------------------------
 ISODUCT   Hot Duct
 Duct wavvec =(   0.549    ,  -2.010E-03) m^-1
 Heat extracted:    10.5     Watts
   557.7     72589.       6.9     -0.00032 -0.08748         -11.76    -11.76
!----------------------------------  3 ----------------------------------
 HXFRST    Hot HX
 Heat exch wavvec =(   0.669    ,  -0.194    ) m^-1
 Heat =    2210.200 (W)    metal temp=     563.295 Kelvin
   557.7     71424.     482.4     -0.00202 -0.09651        2198.44    -95.37
!----------------------------------  4 ----------------------------------
 STKCIRC   Honey Stack
   306.4     65548.    3147.5      0.01282 -0.15903        2198.44    169.94
!----------------------------------  5 ----------------------------------
 HXLAST    Cold HX
 Heat exch wavvec =(   0.858    ,  -0.162    ) m^-1
 Heat =   -2113.895 (W)    metal temp=     303.000 Kelvin
   306.4     62913.    3568.5      0.01215 -0.16675          84.55     84.55
!----------------------------------  6 ----------------------------------
 ISODUCT   Cold Duct
 Duct wavvec =(   0.740    ,  -1.647E-03) m^-1
 Heat extracted:    83.9     Watts
   306.4    -69442.   -4136.6     -0.00002  0.00000           0.64      0.64
!----------------------------------  7 ----------------------------------
 ENDCAP    Cold End
 Heat extracted:   0.642     Watts
   306.4    -69442.   -4136.6      0.00000  0.00000           0.00      0.00
!----------------------------------  8 ----------------------------------
 HARDEND
 inverse impedance (rho a U/p A)=(  -3.410E-12,   2.163E-10)

   306.4    -69442.   -4136.6      0.00000  0.00000           0.00      0.00
```

This run will also produce the following `.out` file:

```
TITLE     Five-Inch Thermoacoustic Engine
!--------------------------------  0 --------------------------------
 BEGIN      Initial
  1.3800E+06 a Mean P    Pa               121.0     A Freq.  G( 0b)     P
   121.0     b Freq.     Hz     G          557.7     B T-beg  G( 0c)     P
   557.7     c T-beg     K      G        7.3450E+04 C |p|@0  G( 0d)     P
  7.3450E+04 d |p|@0     Pa     G
  0.0000     e Ph(p)0    deg
  0.0000     f |U|@0    m^3/s
  0.0000     g Ph(U)0    deg
 helium     Gas type
 ideal      Solid type
!--------------------------------  1 --------------------------------
 ENDCAP     Hot End
  1.2920E-02 a Area      m^2            7.3450E+04 A |p|       Pa
                                        0.0000     B Ph(p)     deg
                                        3.3241E-05 C |U|      m^3/s
                                         180.0     D Ph(U)     deg
                                        -1.221     E Hdot      W
 sameas  0  Gas type                    -1.221     F Work      W
 ideal      Solid type                  -1.221     G HeatIn    W
```

```
!--------------------------------  2 --------------------------------
 ISODUCT    Hot Duct
 sameas  1a  a Area      m^2             7.2589E+04 A |p|       Pa
  0.4030     b Perim      m               5.4761E-03 B Ph(p)     deg
  0.2790     c Length     m               8.7480E-02 C |U|     m^3/s
                                         -90.21     D Ph(U)     deg
                                         -11.76     E Hdot      W
 sameas  0  Gas type                     -11.76     F Work      W
 ideal      Solid type                   -10.54     G HeatIn    W
!--------------------------------  3 --------------------------------
 HXFRST     Hot HX
 sameas  1a  a Area      m^2             7.1425E+04 A |p|       Pa
  0.3930     b GasA/A                     0.3869     B Ph(p)     deg
  6.0000E-02 c Length     m               9.6528E-02 C |U|     m^3/s
  4.8300E-04 d y0         m              -91.20     D Ph(U)     deg
   2210.     e HeatIn     W               2198.     E Hdot      W
   550.0     f Est-T      K     (t)      -95.37     F Work      W
 sameas  0  Gas type                      2210.     G Heat       W
 ideal      Solid type                    563.3     H MetalT     K
!--------------------------------  4 --------------------------------
 STKCIRC    Honey Stack
 sameas  1a  a Area      m^2             6.5624E+04 A |p|       Pa
  0.8100     b GasA/A                     2.749     B Ph(p)     deg
  0.2790     c Length     m               0.1595     C |U|     m^3/s
  5.0000E-04 d radius     m              -85.39     D Ph(U)     deg
  5.0000E-05 e Lplate     m               2198.     E Hdot      W
                                          169.9     F Work      W
                                          557.7     G T-beg      K
 helium     Gas type                     306.4     H T-end      K
 stainless  Solid type                   265.3     I StkWrk     W
!--------------------------------  5 --------------------------------
 HXLAST     Cold HX
  1.2670E-02 a Area      m^2             6.3014E+04 A |p|       Pa
  0.4860     b GasA/A                     3.246     B Ph(p)     deg
  5.0800E-02 c Length     m               0.1672     C |U|     m^3/s
  4.0600E-04 d y0         m              -85.83     D Ph(U)     deg
  0.0000     e HeatIn     W     (t)       84.55     E Hdot      W
   303.0     f Est-T      K    = 5H?      84.55     F Work      W
 helium     Gas type                    -2114.     G Heat       W
 ideal      Solid type                    303.0     H MetalT     K
!--------------------------------  6 --------------------------------
 ISODUCT    Cold Duct
 sameas  5a  a Area      m^2             6.9565E+04 A |p|       Pa
  0.3990     b Perim      m              -176.6     B Ph(p)     deg
   3.654     c Length     m               1.8467E-05 C |U|     m^3/s
                                         -176.6     D Ph(U)     deg
                                          0.6423    E Hdot      W
 helium     Gas type                     0.6423     F Work      W
 ideal      Solid type                  -83.90     G HeatIn    W
!--------------------------------  7 --------------------------------
 ENDCAP     Cold End
 sameas  5a  a Area      m^2             6.9565E+04 A |p|       Pa
                                         -176.6     B Ph(p)     deg
                                          8.5383E-11 C |U|     m^3/s
                                         -85.69     D Ph(U)     deg
                                        -4.6811E-08 E Hdot      W
 helium     Gas type                    -4.6811E-08 F Work      W
 ideal      Solid type                  -0.6423     G HeatIn    W
```

```
!--------------------------------  8 ---------------------------------
 HARDEND
  0.0000     a R(1/Z)          = 8G?     6.9565E+04 A |p|       Pa
  0.0000     b I(1/Z)          = 8H?     -176.6     B Ph(p)     deg
                                         8.5383E-11 C |U|     m^3/s
                                         -85.69     D Ph(U)     deg
                                        -4.6811E-08 E Hdot      W
                                        -4.6811E-08 F Work      W
                                        -3.4102E-12 G R(1/Z)
 helium     Gas type                    2.1633E-10 H I(1/Z)
 ideal      Solid type                   306.4     I   T       K

! The restart information below was generated by a previous run
! You may wish to delete this information before starting a run
! where you will (interactively) specify a different iteration
! mode.  Edit this data only if you really know your model!
INVARS      3  0  2  0  3  0  4
TARGS       3  5  6  8  1  8  2
SPECIALS    0
```

The `.dat` file is a segment-by-segment listing of results of the run. The three members of the guess vector (f, T_{begin}, and $|p_1|_{\text{begin}}$), which we had guessed would be near 100 Hz, 500 Kelvin, and 80,000 Pa, have turned out to be 121.020 Hz, 557.6 Kelvin, and 73,419 Pa; these values appear in the first few lines of `5inch.dat`. Temperature; real and imaginary pressure and volume velocity; energy flow; and work flow are listed at each transition between segments. Be sure the complex volume velocity at `HARDEnd` is zero, as required by two members of the target vector.

Some segments have additional information listed. Ducts and heat exchangers list wavevector (mostly real in the wide-open ducts; with large imaginary components in the much more lossy heat exchangers). Heat exchangers also list heat flow and metal temperature. Note that the metal is hotter than the gas in the hot heat exchanger, where the (positive) heat flows from metal to gas, and that the metal is cooler than the gas in the cold heat exchanger, where the (negative) heat flow is from gas to metal. Note also that DeltaE successfully hit the target metal temperature of 303 Kelvin in the cold heat exchanger.

Now examine the energy and work flow columns in `5inch.dat`. The hot endcap absorbs 1.2 W of work, and the hot duct absorbs $11.7 - 1.2 = 10.5$ W of work. The minus signs on energy and work indicate energy flows 'up' the apparatus, toward the `BEGINning`.

The hot heat exchanger absorbs $95.28 - 11.75 = 83.53$ W of work. Because 2210.2 W of heat are added through it, the energy flow must increase by that amount; hence, the energy flow changes from -11.7 W to 2198.5 W in the hot heat exchanger.

The energy flow remains constant at 2198.5 W through the stack, which produces $169.81 - (-95.28) = 265.09$ W of work. Part of that work (95.28 W) flows up to supply

work to the hot parts of the engine; the rest (169.81 W) flows down to supply work to the cold parts of the engine.

An examination of the cold heat exchanger listing parallels that of the hot heat exchanger, and the cold duct and endcap parallel the hot ones.

Some of this information is also available in the `.out` file, where it appears in a format that can be used as an input file for subsequent runs. The `.out` file is also a segment-by-segment listing, with a restart table appended. In the segment-by segment listing, the variables on the left are used in the input file. They include anything that can be used as a guess or target. Anything that was used as guess or independent plot variable contains its most recent value instead of the initial value supplied be the `.in` file. The variables on the right can be used as dependent variables in plots and can be compared to targets. We will encounter examples of each as we examine typical segments of this file.

The left portion of the `BEGIN` segment is in the `.in`-file format. `Freq, T-beg`, and `|p|@0` are marked with "`G`" signifying their membership in the guess vector. They also appear in the right column, marked with "`P`", signifying their status as default dependent plot variables. The right column of the `BEGIN` segment is a special case: it contains a copy of each guess vector variable with the values that were used in DeltaE's last iteration. To identify their origin, the units for each of these 'output' variables are replaced by the address (e.g., "`0b`") that they were copied from. This occurs *only* in the `BEGIN` segment.

Now examine the cold heat-exchanger segment. Again, the left column is the familiar input-file format. HeatIn is marked "`(t)`" to indicate that it is a potential target variable, though we did not use it as such. `Est-T` is marked "`=5H?`" to show that it is indeed a target variable, to be compared to the computed MetalT variable that appears in the right column. For all input (left side) parameters that DeltaE recognizes as potential targets, it knows the location of the appropriate result to compare with the target value. (The set of *freetarget* segments, each of which introduces a new target variable, allows the result to which it is compared to be flexibly defined. See Section V.A for an introduction to freetargets.)

`HARDEnd` has two more examples of the markers that indicate target variables. There, the target values are 0.0, and DeltaE's solution has reached 1.408e-6 and 5.303e-5, which it judges to be close enough to zero.

The restart table at the end is translated thus:

```
INVARS      3  0  2  0  3  0  4    means 3 variables: 0b, 0c, 0d
TARGS       3  5  6  8  1  8  2    means 3 variables: 5f, 8a, 8b
```

This is an encoded version of the same information that is indicated by the guess and target flags, explained above, and is visible in the vector status summary table. Here, DELTAE would find this information automatically when using this `.out` file as a new input file.

To plot some results for this 5-inch engine case, execute DELTAE with this file again and modify the Plot summary to be

```
Dependent Variables (outputs):
PLOTS       0A          0B          0C          3H         8A
 name  BEGIN:Freq. BEGIN:T-beg BEGIN:|p|@0 HXFRS:Metal HARDE:|p|
 units      Hz          K           Pa           K         Pa
Indpendent Variables (inputs):
 Outer loop:   3e  HXFRS:HeatI Beg=  9.50E+02 End=   50.     Step=  -33.
```

Accomplishing this process required that we "plot another parameter" three times to add 3H and 8A to the dependent variable list and establish `3e` as independent variable and set its initial, final, and step values. (`T-beg` and `|p|@0` are of minor interest now, but could not be deleted from the list of plot variables because members of the guess vector appear here by default.)

Next, we modified mean pressure to be 19.2 bar, and ran the code. When completed, we modified mean pressure to 13.8 bar, and ran it again, appending the new results to the `.plt` file. Three more runs with mean pressures of 9.6, 6.9, and 5.2 bar completed the data set. We exited from DELTAE, and checked to see that it has created the `.des` and `.plt` files:

```
HXFRS:HeatI BEGIN:Freq. BEGIN:T-beg BEGIN:|p|@0 HXFRS:Metal HARDE:|p|
      W          Hz          K           Pa           K         Pa
     3e          0A          0B          0C          3H         8A

    950.0       120.6       562.7      5.9741E+04   566.2      5.6827E+04
    916.7       120.5       562.6      5.8637E+04   566.1      5.5777E+04
    883.4       120.5       562.6      5.7511E+04   566.0      5.4706E+04
    850.1       120.5       562.5      5.6362E+04   565.9      5.3614E+04
    816.8       120.5       562.5      5.5190E+04   565.7      5.2499E+04
      ⋮
```

We read this `.plt` file into a spreadsheet/graphics program for minimal massaging: convert pressure amplitude at the cold end from Pascals to bar, and then square that number; subtract 303 Kelvin from T_h, and add the heat leak to the room to Q_h. Plotting these results then yields the curves shown in Fig. III.2, resembling Figs. 5, 6, and 7 in *J. Acoust. Soc. Am.* **92**, 1551 (1992). These curves differ slightly from those in the article, because of the inclusion of the small gas-to-metal temperature differences in the heat exchangers.

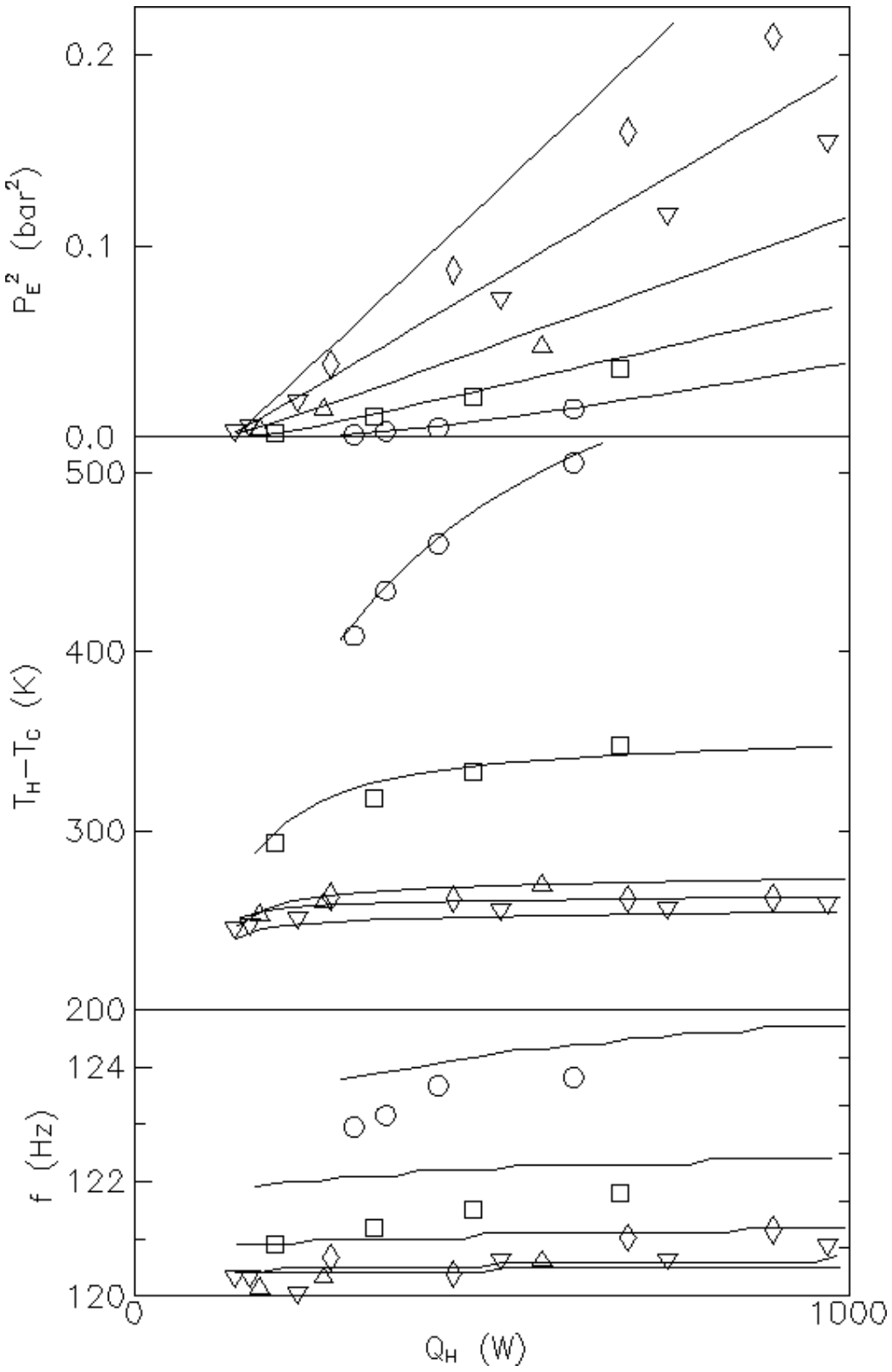


Figure III.2: 5-inch engine results. Lines are DeltaE results; points are from experimental data.

More detailed comparison between DELTAE computations and measurements with this apparatus can be found in *J. Acoust. Soc. Am.* **95**, 1405 (1994).

C Hofler's Thermoacoustic Refrigerator

Tom Hofler's thermoacoustic refrigerator was described in detail in his Ph. D. thesis "Thermoacoustic Refrigerator Design and Performance," UC San Diego, Physics Department (1986). The work was also summarized in the proceedings of the 5th International Cryocoolers Conference, 1988, Monterey CA, p. 93. We use this case to further illustrate capabilities of DELTAE, generating curves similar to Figs. 16 and 17 in Hofler's thesis (Figs. 5 and 6 in the Cryocoolers proceedings).

The apparatus is shown in Fig. III.3:

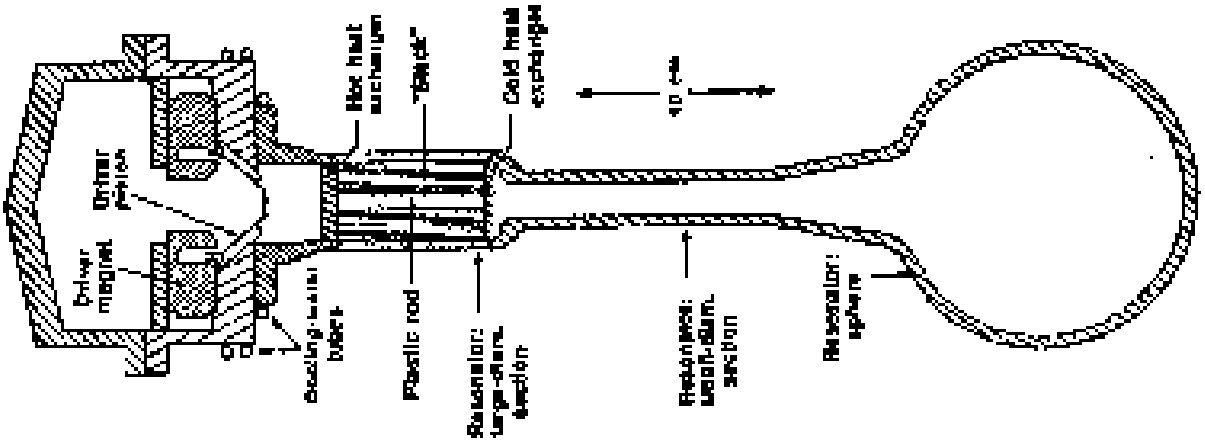


Figure III.3: Hofler's thermoacoustic refrigerator.

We began with an input file (`hofler.in`, in the `examples` directory) whose geometry is that of Hofler's 'long' apparatus:

```
TITLE     Hofler's 1986 thermoacoustic refrigerator

! Geometry comes from Hofler thesis, pages 28, 64, 68, 115, 130, 133.

BEGIN
1.0e6 Pa     Mean P
500. Hz      Freq.
300. K       T-beg
3.0e4 Pa     |p|@0
0.0 deg      Ph(p)0
```

```
5.0e-4 m3/s |U|@0
0.000 deg   Ph(U)0
helium      Gas

ENDCAP       driver end                1
1.134e-3 m2  Area
SAMEAS 0     Gas

ISODUCT    room temp duct              2
SAMEAS 1   Area
0.119 m    Perim
4.26e-2 m  Length
SAMEAS 0   Gas

HXFRST    room temp heat exchanger     3
SAMEAS 1  Area
0.600     GasA/A
6.35e-3 m Length
1.9e-4 m  y0
-20.0 W   HeatIn
300. K    Est-T  (I hope this was the experimental value.)
SAMEAS 0  Gas

STKSLAB   Stack                        4
SAMEAS 1  Area
0.724     GasA/A
7.85e-2 m Length
1.8e-4 m  y0
4.0e-5 m  Lplate
SAMEAS 0  Gas
kapton    Solid

HXLAST    Cold heat exchanger          5
SAMEAS 1  Area
0.67      GasA/A
2.54e-3 m Length
2.55e-4 m y0
3.0 W     Heatin
200. K    Est-T
SAMEAS 0  Gas

ISODUCT     Cold Duct                  6
3.84e-4 m2  Area
0.0694 m    Perim
0.167 m     Length
SAMEAS 0    Gas

ISOCONE                                7
SAMEAS 6   Initial Area
SAMEAS 6   In Perim
6.68e-2 m  Length
1.16e-3 m2 Final area
0.121 m    Final perim
SAMEAS 0

COMPLIANCE  end bulb                   8
0.049 m2    Area
1.06e-3 m3  Volume
```

```
SAMEAS 0    Gas

HARDEND
0.000     R(Zin)
0.000     I(Zin)
SAMEAS 0  Gas type
```

Note the use of segment type `STKSLab` to model the parallel-plate stack geometry, and the use of segment types `ISOCOne` and `COMPLiance` to model parts of the cold portion of the resonator.

Executing DeltaE and choosing this input file, we used `(C)lear|set` to ask for default targets,

```
No vectors defined...do you want enable a default
set of targets&guesses for this model? (y/n)   y
Is this a prime-mover or a heat pump(p|h)?  h
```

responding with '`h`' because we now have a heat pump, and examine the vector status summary:

```
Iteration Vectors Summary:
 GUESS      0b          0c          0f
 name  BEGIN:Freq. BEGIN:T-beg BEGIN:|U|@0
 value    5.00E+02    3.00E+02    5.00E-04
 units     Hz           K          m^3/s
TARGET      3f          9a          9b
 name  HXFRS:Est-T HARDE:R(1/Z HARDE:I(1/Z
 units      K
 value    3.00E+02    0.00        0.00

Potential TARGETS still available:
Addr Seg:Par-Type    Current Value
 5e HXLAST:HeatIn=   3.000         W
 5f HXLAST:Est-T =   200.0         K
```

This time we are not satisfied with DeltaE's default choice of elements of this table. We would like to generate a curve like Hofler's Fig. 16. To show off DeltaE's ability to handle more dimensions in its shooting-method algorithm, and to get a direct grip on the independent variable in Hofler's figure, we made the refrigeration power a target, adding the room temperature waste heat to the guess vector. After making these changes, the vector summary looks like

```
Iteration Vectors Summary:
 GUESS      0b          0c          0f          3e
 name  BEGIN:Freq. BEGIN:T-beg BEGIN:|U|@0 HXFRS:HeatI
```

```
 value    5.00E+02    3.00E+02    5.00E-04    -20.0
 units     Hz          K          m^3/s         W
TARGET      3f          5e          9a          9b
 name  HXFRS:Est-T HXLAS:HeatI HARDE:R(1/Z HARDE:I(1/Z
 units      K           W
 value    3.00E+02     3.0         .00         .00

Potential TARGETS still available are:
Addr Seg:Par-Type    Current Value
 5f HXLAST:Est-T =   200.0         K
```

Running this case produced the following .DAT file:

```
 -= Hofler's 1986 thermoacoustic refrigerator                                =-
 frequency=     499.165Hz     mean pressure=   1.000E+06Pa

  T(K)          p(Pa)               U(m^3/s)             hdot(W)   wdot(W)
   300.7     30000.       0.0      0.00051  0.00000           7.66      7.66
!----------------------------------  1 ----------------------------------
 ENDCAP    Driver end       1
 Heat extracted:   3.464E-02 Watts
   300.7     30000.       0.0      0.00051  0.00000           7.62      7.62
!----------------------------------  2 ----------------------------------
 ISODUCT   Room temp duct
 Duct wavvec =(    3.09    ,  -1.301E-02) m^-1
 Heat extracted:   0.153     Watts
   300.7     29740.     -93.8      0.00049 -0.00273           7.47      7.47
!----------------------------------  3 ----------------------------------
 HXFRST    Room temp duct heat
 Heat exch wavvec =(    3.67    ,  -0.890    ) m^-1
 Heat =      -9.640 (W)    metal temp=     300.000 Kelvin
   300.7     29573.     -69.3      0.00044 -0.00302          -2.17      6.66
!----------------------------------  4 ----------------------------------
 STKSLAB   Stack       4
   218.6     26127.     640.8      0.00025 -0.00678          -2.17      1.05
!----------------------------------  5 ----------------------------------
 HXLAST    Cold HX       5
 Heat exch wavvec =(    4.03    ,  -0.505    ) m^-1
 Heat =       3.000 (W)    metal temp=     218.857 Kelvin
   218.6     25948.     662.0      0.00024 -0.00689           0.83      0.83
!----------------------------------  6 ----------------------------------
 ISODUCT   Cold Duct       6
 Duct wavvec =(    3.63    ,  -2.005E-02) m^-1
 Heat extracted:   0.678     Watts
   218.6      1754.      21.0      0.00027 -0.00862           0.15      0.15
!----------------------------------  7 ----------------------------------
 ISOCONE   7
 Heat extracted:   0.127     Watts
   218.6     -4216.    -138.7      0.00027 -0.00841           0.02      0.02
!----------------------------------  8 ----------------------------------
 COMPLIAN  End Bulb          8
 Heat extracted:   2.251E-02 Watts
   218.6     -4216.    -138.7      0.00000  0.00000           0.00      0.00
!----------------------------------  9 ----------------------------------
 HARDEND   9
 inverse impedance (rho a U/p A)=(   3.213E-10,   1.141E-09)
```

```
   218.6     -4216.    -138.7      0.00000  0.00000           0.00      0.00
```

Close examination of this result for reasonableness reveals a problem: The stack is pumping 2.2 W of energy uphill, but 3.0 W of heat is being removed from the cold heat exchanger! How can this be? The problem is in our use of `ISODUct` and `ISOCOne` in the cold portion of the apparatus. DeltaE assumes that these segments are held isothermal by external means. In this case, that means that in the duct, cone, and compliance, where 0.82 W of work is dissipated into heat, some external means removes that heat. In Hofler's work, that external means was a good thermal connection between these parts and the cold heat exchanger, so that this heat appeared as a load on the cold heat exchanger.

There are two ways to deal with this problem. The first is to simply subtract the 0.82 W from the 3 W when we want to know the "actual" net refrigeration power available at the cold heat exchanger. This is not very elegant. The second is to use the insulated segment types `INSDUct` and `INSCOne`. (DeltaE will insulate the compliance as well.) This will force heat dissipated in these segments to show up in the nearest heat exchanger.

To do this, we use a text editor to edit the input file, changing the two segments from `ISO-` to `INS-`. We also chose to cut the cold heat exchanger heat from 3 W to 2.18 W so that the "true" cooling power would be the same as above. Running this case produced

```
 -= Hofler's 1986 thermoacoustic refrigerator                              =-
frequency=     499.211Hz     mean pressure=   1.000E+06Pa

 T(K)          p(Pa)               U(m^3/s)             hdot(W)   wdot(W)
  300.7     30000.       0.0      0.00051  0.00000           7.66      7.66
!----------------------------------  1 ----------------------------------
ENDCAP    Driver end       1
Heat extracted:   3.464E-02 Watts
  300.7     30000.       0.0      0.00051  0.00000           7.63      7.63
!----------------------------------  2 ----------------------------------
ISODUCT   Room temp duct
Duct wavvec =(    3.09    ,  -1.301E-02) m^-1
Heat extracted:   0.153     Watts
  300.7     29740.     -93.9      0.00049 -0.00273           7.47      7.47
!----------------------------------  3 ----------------------------------
HXFRST    Room temp duct heat
Heat exch wavvec =(    3.67    ,  -0.890    ) m^-1
Heat =      -9.652 (W)    metal temp=     300.000 Kelvin
  300.7     29573.     -69.4      0.00044 -0.00302          -2.18      6.66
!----------------------------------  4 ----------------------------------
STKSLAB   Stack       4
  218.6     26127.     640.8      0.00025 -0.00679          -2.18      1.05
!----------------------------------  5 ----------------------------------
HXLAST    Cold HX       5
Heat exch wavvec =(    4.03    ,  -0.506    ) m^-1
Heat =       2.180 (W)    metal temp=     218.828 Kelvin
  218.6     25948.     662.0      0.00024 -0.00689           0.00      0.83
!----------------------------------  6 ----------------------------------
```

```
 INSDUCT   Cold Duct       6
 Duct wavvec =(    3.63    ,  -2.006E-02) m^-1
 Heat extracted:   0.000     Watts
   218.6      1754.      21.0      0.00027 -0.00862           0.00      0.15
!-----------------------------------  7 -----------------------------------
 INSCONE   7
 Heat extracted:   0.000     Watts
   218.6     -4216.    -138.7      0.00027 -0.00841           0.00      0.02
!-----------------------------------  8 -----------------------------------
 COMPLIAN  End Bulb          8
 Heat extracted:   1.523E-09 Watts
   218.6     -4216.    -138.7      0.00000  0.00000           0.00      0.00
!-----------------------------------  9 -----------------------------------
 HARDEND   9
 inverse impedance (rho a U/p A)=(  -6.693E-12,   1.482E-12)

   218.6     -4216.    -138.7      0.00000  0.00000           0.00      0.00
```

Thus, the work dissipated in the cold portion showed up automatically in the cold heat exchanger.

(INSulated segments are still under development and they don't always do what we want them to do. When using them, carefully examine the results for reasonableness. See Chapter VI for details.)

To generate plots for comparison to Hofler's data, we return to ISODUct and ISOCOne because he added the dissipation in these components to his applied heat load for plotting. We let the heat at the cold heat exchanger be the independent variable, ranging from 2 to 8 W in 0.5 W steps. To plot the temperature ratio and the coefficient of performance (COP) relative to Carnot's COP, we include work at segment 1, T_c, and T_h in the list of plotted variables:

```
Dependent Variables (outputs):
PLOTS     0A          0B           0C          0D           1F          3H
5H
 name BEGIN:Freq. BEGIN:T-beg BEGIN:|U|@0 BEGIN:HeatI ENDCA:Work
HXFRS:Metal HXLAS:
 units    Hz          K          m^3/s         W            W           K
K
Indpendent Variables (inputs):
 Outer loop:   5e  HXLAS:HeatI Beg=   2.0     End=   8.0     Step=  0.50
```

(The first three dependent variables listed are unclearable defaults that we ignore.) After running this case, we changed $|p_1|$ to 0.015 of p_m, changed the range of Q_c to 0.7 to 3.7 W in steps of 0.5 W, and ran it again. Exiting DELTAE, we found the following .des and .plt files for the first case:

```
HXLAS:HeatI BEGIN:Freq .BEGIN:T-beg BEGIN:|U|@0 ENDCA:Work HXFRS:Metal
```

```
HXLAS:Metal
      W         Hz          K          m^3/s        W           K           K
     5e         0A          0B           0C         1F          3H          5H

    2.000      493.1       300.6     4.7359E-04   7.069       300.0       212.9
    2.500      496.2       300.6     4.9203E-04   7.346       300.0       215.9
    3.000      499.2       300.7     5.1039E-04   7.321       300.0       218.9
    3.500      502.1       300.7     5.2869E-04   7.896       300.0       221.8
      ⋮
```

Reading this file into spreadsheet/graphics software, and forming T_c/T_h and COPR yielded the curves in Figs. III.4. These plots come reasonably close to the measurements presented in Figs. 16 and 17 of Hofler's thesis.

Returning to `INS-`, we now use this example to introduce some more new segment types.

Since we frequently find it useful to consider engine efficiency or refrigerator coefficient of performance (COP), normalized by their Carnot values, we have special segments `COPRTarget` and `EFFRTarget` to compute them. We can even use them as targets if desired. We added a `COPRTarget` to our input file:

(skipping the first segments, which we've seen before)

```
!--------------------------------  8 --------------------------------
 COMPLIAN   End Bulb
  4.9000E-02 a Area      m^2              4218.     A |p|       Pa
  1.0600E-03 b Volum     m^3              -178.1    B Ph(p)     deg
                                       1.2793E-10 C |U|     m^3/s
                                          -103.8    D Ph(U)     deg
                                       7.3110E-08 E Hdot      W
 sameas  0  Gas type                   7.3110E-08 F Work      W
 ideal      Solid type                -2.2509E-02 G HeatIn    W
!--------------------------------  9 --------------------------------
 HARDEND
  0.0000     a R(1/Z)          = 9G?     4218.     A |p|       Pa
  0.0000     b I(1/Z)          = 9H?     -178.1    B Ph(p)     deg
                                       1.2793E-10 C |U|     m^3/s
                                          -103.8    D Ph(U)     deg
                                       7.3110E-08 E Hdot      W
                                       7.3110E-08 F Work      W
                                       3.2133E-10 G R(1/Z)
 sameas  0  Gas type                   1.1415E-09 H I(1/Z)
 ideal      Solid type                  218.6     I   T       K

 COPRT      COP/COP-Carnot
 0.13       Target
 5G         NumAdr
 1F         DenomAdr
 3H         ThAdr
 5H         TcAdr
! The restart information below was generated by a previous run
```

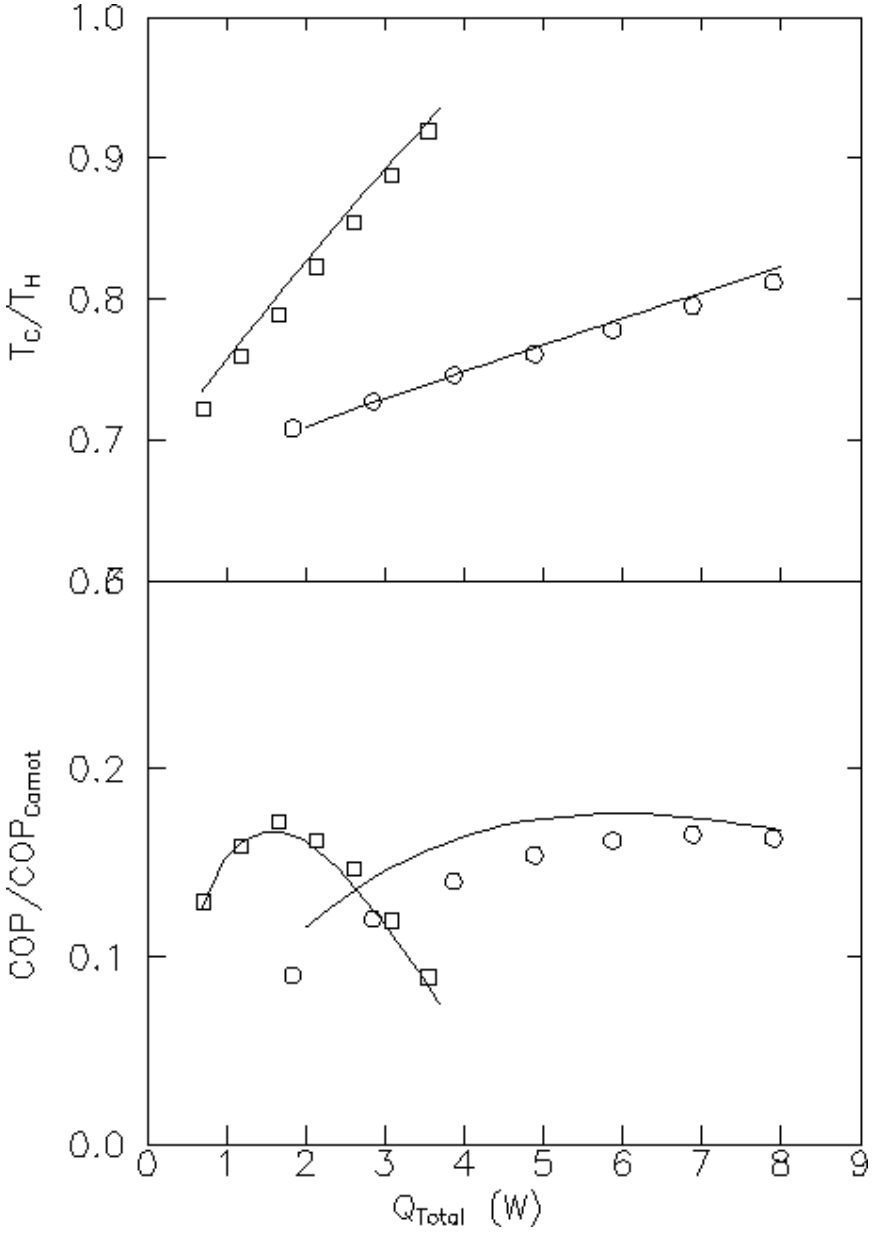


Figure III.4: Hofler refrigerator results. Lines are DELTAE results; points are from experimental data. Squares , $p_1 = 0.015 p_m$. Circles, $p_1 = 0.03 p_m$.

```
! You may wish to delete this information before starting a run
! where you will (interactively) specify a different iteration
! mode.  Edit this data only if you really know your model!
INVARS      4  0  2  0  3  0  6  3  5
TARGS       4  3  6  9  1  9  2 10  1
SPECIALS    0
```

Here, the `COPRTarget` segment directly calculates the result that we calculated laboriously in our spreadsheet/graphics software using data from the `.plt` file. The result appears in the `.out` file and the `.dat` file, and can be tabulated in the `.plt` file.

COPR can be used as a target. For instance, if we change the vector summary to

```
Iteration Vectors Summary:
 GUESS      0b          0c          0f          3e
 name  BEGIN:Freq. BEGIN:T-beg BEGIN:|U|@0 HXFRS:HeatI
 value    4.97E+02    3.01E+02    5.06E-04    -9.6
 units     Hz          K          m^3/s        W
TARGET      3f          9a          9b         10a
 name  HXFRS:Est-T HARDE:R(1/Z HARDE:I(1/Z COPRT:Targe
 units      K
 value    3.00E+02     .00         .00         .13

Potential TARGETS still available are:
Addr Seg:Par-Type    Current Value
 5e HXLAST:HeatIn=   2.190         W
 5f HXLAST:Est-T =   200.0         K
```

and run the code, we can find an operating point on the plots above that corresponds to COPR = 0.13.

Finally, a more realistic driver was added to the system, using `VSPEAKer`. We edited the input file, adding `VSPEAKer` near the beginning. We deleted the `ENDCAp` segment that was near the beginning because `VSPEAker` accounts for the oscillatory pressurization losses on its surface area. We also added a difference target `DIFFTarget` at the end.

```
TITLE     Hofler's 1986 thermoacoustic refrigerator, w speaker

 BEGIN                          0
   1.000E+06 a Mean P    Pa
    500.     b Freq.     Hz
    300.     c T-beg     K
   3.000E+04 d |p|@0     Pa
   150.0     e Ph(p)0    deg
    .000     f |U|@0    m^3/s
    .000     g Ph(U)0    deg
 helium     Gas type
 ideal      Solid type
```

```
VSPEAKER                        1
  6.000E-04 a Area      m^2
   6.00     b   R      ohms
   .000     c L         H
   8.00     d B x L    T-m
  5.000E-03 e   M       kg
   .000     f   K       N/m
   .000     g  Rm     N-s/m
   20.      h AplVol    V
SAMEAS  0  Gas type
ideal      Solid type

ISODUCT    room temp duct       2
  1.134E-03 a Area      m^2
   .119     b Perim      m
  4.260E-02 c Length     m
SAMEAS  0  Gas type
ideal      Solid type

HXFRST     room temp heat excha 3
SAMEAS  2a  a Area      m^2
   .600     b GasA/A
  6.350E-03 c Length     m
  1.900E-04 d y0         m
  -10.      e HeatIn     W
   300.     f Est-T      K
SAMEAS  0  Gas type
ideal      Solid type

STKSLAB    Stack                4
SAMEAS  2a  a Area      m^2
   .724     b GasA/A
  7.850E-02 c Length     m
  1.800E-04 d y0         m
  4.000E-05 e Lplate     m
SAMEAS  0  Gas type
kapton     Solid type

HXLAST     Cold heat exchanger  5
SAMEAS  2a  a Area      m^2
   .670     b GasA/A
  2.540E-03 c Length     m
  2.550E-04 d y0         m
   2.19     e HeatIn     W
   200.     f Est-T      K
SAMEAS  0  Gas type
ideal      Solid type

INSDUCT    Cold duct            6
  3.840E-04 a Area      m^2
  6.940E-02 b Perim      m
   .167     c Length     m
SAMEAS  0  Gas type
ideal      Solid type

INSCONE                         7
SAMEAS  6a  a AreaI     m^2
SAMEAS  6b  b PerimI     m
```

```
  6.680E-02 c Length     m
  1.160E-03 d AreaF     m^2
   .121     e PerimF     m
SAMEAS  0  Gas type
ideal      Solid type

COMPLIANCE end bulb             8
  4.900E-02 a Area      m^2
  1.060E-03 b Volum     m^3
SAMEAS  0  Gas type
ideal      Solid type

HARDEND                         9
   .000     a R(1/Z)
   .000     b I(1/Z)
SAMEAS  0  Gas type
ideal      Solid type

DIFFTARGET                      10
   .000     a Target
 1B b
 1L c
```

The mass, resistance, and force constant for the speaker roughly reflect the values given in Hofler's thesis. We estimate it will take about 20 V to drive it.

We used the difference target `DIFFTarget` segment to maintain resonance, by ensuring that the phases of p_1 and U_1 are equal at the driver. We did this by forcing their difference, computed by subtracting the values addressed by lines `10b` and `10c`, to be zero, the value given in line `10a`. Examination of a `VSPEAKer` segment output

```
VSPEAKER
 6.0000E-04 a Area      m^2            3.0000E+04 A |p|       Pa
  6.000     b   R      ohms             153.8     B Ph(p)     deg
 0.0000     c L         H              5.0874E-04 C |U|     m^3/s
  8.000     d B x L    T-m              153.8     D Ph(U)     deg
 5.0000E-03 e   M      kg               7.631     E Hdot      W
 0.0000     f   K      N/m              7.631     F Work      W
 0.0000     g   Rm   N-s/m              31.17     G WorkIn    W
  22.63     h AplVol    V      G        22.63     H Volts     V
                                        2.800     I Amps      V
                                       -10.30     J Ph(Ze)   deg
                                       5.0996E-04 K  |Ux|  m^3/s
sameas  0  Gas type                     153.8     L Ph(-Ux   deg
ideal      Solid type                  -23.54     M HeatIn    W
```

shows us that lines 10b and c should contain addresses 1B and 1L.

Running DeltaE with this input file, we modified guesses and targets to arrive at

```
Iteration Vectors Summary:
```

```
 GUESS      0b          0c          0e          1h          3e
 name  BEGIN:Freq. BEGIN:T-beg BEGIN:Ph(p) VSPEA:AplVo HXFRS:HeatI
 value    5.00E+02    3.00E+02    150.         20.        -10.0
 units     Hz          K           deg         V           W
TARGET      3f          5e          9a          9b         10a
 name  HXFRS:Est-T HXLAS:HeatI HARDE:R(1/Z HARDE:I(1/Z DIFFT:Targe
 units      K           W
 value    3.00E+02     2.2         .00         .00         .00

Potential TARGETS still available are:
Addr Seg:Par-Type    Current Value
 5f HXLAST:Est-T =   200.0         K
```

This shows our five-dimensional search. It is the most complicated vector summary table we have yet encountered, so we pause to discuss how we chose our vectors. We definitely needed the two `HARDEnd` impedances in the target vector (there is no hole in the end of the apparatus). Experimentally, we maintain the hot heat exchanger at 300 Kelvin; but DeltaE computes that as a result of each integration pass, so it must also be a target. Experimentally, we control the heat load on the cold heat exchanger, but because this is an `HXLASt`, DeltaE calculates it as a result of each integration pass, so it too must be a target. So far we have four targets, so we require four guesses. Look first at the `BEGIN` segment for candidate guesses. Clearly the beginning temperature should be a guess: we need to guess beginning T to arrive at `HXFRSt` T correctly. Next, we must guess the frequency to maintain resonance. But how is resonance determined experimentally? By comparing the phases of p_1 and U_1 at the driver: hence, we added their difference$= 0$ as a fifth target. We need the phase of p_1 at the beginning to be a guess, since the phase of everything is determined relative to that of the speaker voltage phase, which is fixed at 0^o. Other good candidate guesses are heats in `HXFRSt`s or `HXMIDl`s. The heat in the first heat exchanger must be guessed because we don't control it experimentally yet it is required by DeltaE in each pass. By now we have five targets and four guesses; we needed one more guess. Our guess could be $|p_1|$ at the beginning, which would be an experimental result if we controlled the drive voltage. Instead, however, we let the drive voltage be the guess because the experimenter used it to get $|p_1|$ to be 0.03 p_m.

Choosing the vector members is not easy for a complicated thermoacoustic system. To choose them wisely, there is no substitute for careful thought about the system and what you want it to do. We offer a few general guides for this careful thought process. It is helpful to think about what variables are (or could be, in principle) experimentally controlled and what variables are experimentally observed. These must be compared with the variables that DeltaE needs as inputs during each integration pass through the system and those that DeltaE computes as results during each integration pass.

	Experimentally Controlled Variable	Experimentally a Result
Variable needed as input for each pass of DeltaE's integration	simply fixed in input file	guess
Variable computed as result of each pass of DeltaE's integration	target	simply a result in output files

Note that our definition of an experimental result is more general than usual. In the Hofler refrigerator case, we considered the drive voltage an experimental result because it is determined experimentally by the condition that the pressure amplitude have the desired value. The viewpoint expressed in this table is appropriate for comparison of DeltaE and experimental data. In this case, geometrical parameters are simply fixed. Targets are experimentally fixed or controlled variables that are results of a single pass of numerical integration, chosen from among T_m, p_1, and U_1 (everywhere but in `BEGIN`); heats at `HXLAST`s; current magnitudes and phases in `VDUCER`s and voltage magnitudes and phases in `IDUCER`s; etc. Guesses are known or unknown experimental results chosen from among f, the magnitude and phase of U_1-`BEGIN` and p_1-`BEGIN`, T_m-`BEGIN`, heats at `HXFRST`s or `HXMIDL`s, and the magnitude and phase of voltage at `VDUCER`s, etc.

When designing hardware instead of analyzing it, a different viewpoint may be adopted. In this case, many geometrical parameters are not yet fixed, but desired operating temperatures, powers, frequency, etc. have been chosen. Often, several geometrical parameters are included as guesses, and more temperatures and other numerical results are included as targets. Hence, another useful way to think about guesses and targets is represented by the following table:

	Variable *we* want to think of as fixed	Variable *we* want to think of as a result
Variable needed as input for each pass of DELTAE's integration	simply fixed in input file	guess
Variable computed as result of each pass of DELTAE's integration	target	simply a result in output files

Now we return to our example. Running this case produces the following `dat` file:

```
 -= Hofler's 1986 thermoacoustic refrigerator, w speaker                    =-
 frequency=     499.272Hz     mean pressure=   1.000E+06Pa
  T(K)          p(Pa)               U(m^3/s)             hdot(W)   wdot(W)
   300.7    -26913.   13255.7      0.00000  0.00000           0.00      0.00
!----------------------------------  1 ----------------------------------
 VSPEAKER
 (    22.6    ,   0.000    ) Volts,(    2.75    ,   0.501    ) Amps
 Heat extracted: I^2 R=  23.5    , u^2 Rm= 0.000    , B-Layer= 1.833E-02 Watts.
   300.7    -26913.   13255.7     -0.00046  0.00022           7.63      7.63
!----------------------------------  2 ----------------------------------
 ISODUCT   Room temp duct
 Duct wavvec =(    3.09    ,  -1.301E-02) m^-1
 Heat extracted:   0.153     Watts
   300.7    -26638.   13225.3      0.00076  0.00267           7.48      7.48
!----------------------------------  3 ----------------------------------
 HXFRST    Room temp duct heat
 Heat exch wavvec =(    3.67    ,  -0.890    ) m^-1
 Heat =      -9.668 (W)    metal temp=     300.000 Kelvin
   300.7    -26499.   13129.3      0.00094  0.00290          -2.19      6.67
!----------------------------------  4 ----------------------------------
 STKSLAB   Stack
   218.7    -23721.   10969.4      0.00278  0.00620          -2.19      1.05
!----------------------------------  5 ----------------------------------
 HXLAST    Cold HX
 Heat exch wavvec =(    4.03    ,  -0.506    ) m^-1
 Heat =       2.190 (W)    metal temp=     218.888 Kelvin
   218.7    -23570.   10871.3      0.00283  0.00629           0.00      0.83
!----------------------------------  6 ----------------------------------
 INSDUCT   Cold Duct
 Duct wavvec =(    3.63    ,  -2.006E-02) m^-1
 Heat extracted:   0.000     Watts
   218.7     -1583.     756.3      0.00357  0.00786           0.00      0.15
!----------------------------------  7 ----------------------------------
 INSCONE
 Heat extracted:   0.000     Watts
   218.7      3843.   -1738.5      0.00348  0.00766           0.00      0.02
!----------------------------------  8 ----------------------------------
 COMPLIAN  End Bulb
```

```
 Heat extracted:   3.051E-05 Watts
   218.7      3843.   -1738.5      0.00000  0.00000           0.00      0.00
!----------------------------------  9 ----------------------------------
 HARDEND
 inverse impedance (rho a U/p A)=(  -1.341E-07,   2.214E-07)
   218.7      3843.   -1738.5      0.00000  0.00000           0.00      0.00
!---------------------------------- 10 ----------------------------------
 DIFFTARGET
 Derived difference =   8.269E-06
   218.7      3843.   -1738.5      0.00000  0.00000           0.00      0.00
```

This run also produces the following `.out` file:

```
TITLE     Hofler's 1986 thermoacoustic refrigerator, w speaker
!---------------------------------  0 ---------------------------------
 BEGIN
  1.0000E+06 a Mean P     Pa               499.3     A Freq.  G( 0b)     P
   499.3     b Freq.      Hz      G        300.7     B T-beg  G( 0c)     P
   300.7     c T-beg      K       G        153.8     C Ph(p)0 G( 0e)     P
  3.0000E+04 d |p|@0      Pa               22.63     D AplVol G( 1h)     P
   153.8     e Ph(p)0     deg     G       -9.668     E HeatIn G( 3e)     P
  0.0000     f |U|@0     m^3/s
  0.0000     g Ph(U)0     deg
 helium      Gas type
 ideal       Solid type
!---------------------------------  1 ---------------------------------
 VSPEAKER
  6.0000E-04 a Area       m^2           3.0000E+04 A |p|        Pa
   6.000     b   R        ohms           153.8     B Ph(p)      deg
  0.0000     c L           H             5.0874E-04 C |U|      m^3/s
   8.000     d B x L     T-m              153.8     D Ph(U)      deg
  5.0000E-03 e   M        kg              7.631     E Hdot       W
  0.0000     f   K        N/m             7.631     F Work       W
  0.0000     g   Rm    N-s/m              31.17     G WorkIn     W
   22.63     h AplVol     V       G        22.63     H Volts      V
                                          2.800     I Amps       V
                                         -10.30     J Ph(Ze)    deg
                                         5.0996E-04 K  |Ux|   m^3/s
 sameas  0  Gas type                      153.8     L Ph(-Ux    deg
 ideal      Solid type                   -23.54     M HeatIn     W
!---------------------------------  2 ---------------------------------
 ISODUCT    Room temp duct
  1.1340E-03 a Area       m^2           2.9741E+04 A |p|        Pa
  0.1190     b Perim       m             153.6     B Ph(p)      deg
  4.2600E-02 c Length      m             2.7746E-03 C |U|      m^3/s
                                          74.04     D Ph(U)      deg
                                          7.478     E Hdot       W
 sameas  0  Gas type                      7.478     F Work       W
 ideal      Solid type                   -0.1534     G HeatIn     W
!---------------------------------  3 ---------------------------------
 HXFRST     Room temp duct heat
 sameas  2a  a Area       m^2           2.9573E+04 A |p|        Pa
  0.6000     b GasA/A                    153.6     B Ph(p)      deg
  6.3500E-03 c Length      m             3.0511E-03 C |U|      m^3/s
  1.9000E-04 d y0          m              72.15     D Ph(U)      deg
  -9.668     e HeatIn      W     G       -2.190     E Hdot       W
```

```
  300.0     f Est-T      K    = 3H?      6.670     F Work      W
sameas  0  Gas type                    -9.668     G Heat       W
ideal      Solid type                   300.0     H MetalT     K
!--------------------------------  4 --------------------------------
STKSLAB    Stack
sameas  2a  a Area      m^2             2.6134E+04 A |p|       Pa
 0.7240     b GasA/A                     155.2     B Ph(p)     deg
 7.8500E-02 c Length     m              6.7906E-03 C |U|     m^3/s
 1.8000E-04 d y0         m               65.86     D Ph(U)     deg
 4.0000E-05 e Lplate     m              -2.190     E Hdot      W
                                         1.054     F Work      W
                                         300.7     G T-beg      K
sameas  0  Gas type                      218.7     H T-end      K
kapton     Solid type                   -5.616     I StkWrk     W
!--------------------------------  5 --------------------------------
HXLAST     Cold HX
sameas  2a  a Area      m^2             2.5956E+04 A |p|       Pa
 0.6700     b GasA/A                     155.2     B Ph(p)     deg
 2.5400E-03 c Length     m              6.8953E-03 C |U|     m^3/s
 2.5500E-04 d y0         m               65.77     D Ph(U)     deg
  2.190     e HeatIn     W    = 5G?      0.0000     E Hdot      W
  200.0     f Est-T      K     (t)       0.8280     F Work      W
sameas  0  Gas type                      2.190     G Heat       W
ideal      Solid type                    218.9     H MetalT     K
!--------------------------------  6 --------------------------------
INSDUCT    Cold Duct
 3.8400E-04 a Area      m^2              1754.     A |p|       Pa
 6.9400E-02 b Perim      m               154.5     B Ph(p)     deg
 0.1670     c Length     m              8.6291E-03 C |U|     m^3/s
                                         65.59     D Ph(U)     deg
                                        0.0000     E Hdot      W
sameas  0  Gas type                     0.1494     F Work      W
ideal      Solid type                   0.0000     G HeatIn    W
!--------------------------------  7 --------------------------------
INSCONE
sameas  6a  a AreaI     m^2              4218.     A |p|       Pa
sameas  6b  b PerimI     m              -24.34     B Ph(p)     deg
 6.6800E-02 c Length     m              8.4160E-03 C |U|     m^3/s
 1.1600E-03 d AreaF      m^2             65.59     D Ph(U)     deg
 0.1210     e PerimF     m              0.0000     E Hdot      W
sameas  0  Gas type                     2.2491E-02 F Work      W
ideal      Solid type                   0.0000     G HeatIn    W
!--------------------------------  8 --------------------------------
COMPLIAN   End Bulb
 4.9000E-02 a Area      m^2              4218.     A |p|       Pa
 1.0600E-03 b Volum     m^3             -24.34     B Ph(p)     deg
                                        2.7931E-08 C |U|     m^3/s
                                         96.85     D Ph(U)     deg
                                       -3.0510E-05 E Hdot      W
sameas  0  Gas type                    -3.0510E-05 F Work      W
ideal      Solid type                  -3.0510E-05 G HeatIn    W
!--------------------------------  9 --------------------------------
HARDEND
 0.0000     a R(1/Z)          = 9G?      4218.     A |p|       Pa
 0.0000     b I(1/Z)          = 9H?     -24.34     B Ph(p)     deg
                                        2.7931E-08 C |U|     m^3/s
                                         96.85     D Ph(U)     deg
                                       -3.0510E-05 E Hdot      W
```

```
                                        -3.0510E-05 F Work      W
                                        -1.3406E-07 G R(1/Z)
 sameas  0  Gas type                     2.2143E-07 H I(1/Z)
 ideal      Solid type                     218.7     I   T       K
!-------------------------------- 10 ---------------------------------
 DIFFTARGET
  0.0000     a TargDi          =10A?     8.2692E-06 A D1-D2
  1B b D1Addr
  1L c D2Addr

! The restart information below was generated by a previous run
! You may wish to delete this information before starting a run
! where you will (interactively) specify a different iteration
! mode.  Edit this table only if you really know your model!
INVARS      5  0  2  0  3  0  5  1  8  3  5
TARGS       5  3  6  5  5  9  1  9  2 10  1
SPECIALS    0
```

These acoustic and thermal results are the same as for without the speaker, except that everything is shifted in phase by -26 degrees. This shift occurred because we had set the phase of U_1 at the driver, arbitrarily, at zero before, but now the phase of the speaker voltage determines the zero of phase for the system, and the nonzero imaginary part of its mechanical impedance causes a phase shift between the voltage and the velocity. New results appear in the `VSPEAker` segment; note for example that $|I|^2R/2$ is the difference between the work into the segment ($1/2\Re(I\tilde{V})$) and the work out of it.

D Further Thermoacoustic Features

In this section we list the commonly used thermoacoustic segment types. More details on each can be found in Chapter VI; a complete list can be found in Chapter VI.

`STKCIrc` A stack with circular pores. We use this to model hexagonal honeycomb stacks.

`STKSLab` A stack with parallel-plate geometry.

`STKREct` A stack with rectangular (box) pore geometry.

`STKPIns` A stack comprised of an array of pins parallel to x.

`STKDUct` A stack with lateral dimensions much larger than δ_κ, computed in boundary-layer approximation.

`STKSCreen` A screen regenerator for Stirling systems.

`HXFRSt` A parallel-plate heat exchanger that comes before one of the STK segment types.

`HXMIDl` A parallel-plate heat exchanger that comes between STK segments.

`HXLASt` A parallel-plate heat exchanger that comes after a STK segment.

`TXFRSt, TXMIDl, TXLASt` Tube-array heat exchangers, with the thermoacoustic working fluid inside the tubes.

`SXFRSt, SXMIDl, SXLASt` Stacked-screen heat exchangers, valid only for δ_κ greater than hydraulic radius.

`INSDUct` An insulated duct; the work dissipated in it shows up in the nearest heat exchanger.

`INSCOne` An insulated cone; the work dissipated in it shows up in the nearest heat exchanger.

E Advanced Operations

These menu options are not necessary for ordinary operation of the code, but they offer some substantial conveniences for experienced users.

`(R)estore vectors.` Before beginning iterations during a `(r)un` operation, DELTAE saves copies of the guess vector values. Whenever an unsuccessful run overwrites the guess vector (leaving you and DELTAE hopelessly lost), you can use this option to restore all the parameters that were changed to their starting point. Simply `(R)estore`, modify some value(s), and try again. There are warnings about trying to use this option after the vector table has been edited, which of course would make no sense.

If you do not respond '`y`'es to the prompt about vector restoration and you have one or both plot loops enabled, you will be given an additional option:

```
Restore to state before last (B)egin or (r)un (y|n)? n

Restore from a recently plotted point? y
```

DELTAE will now proceed to display the `.plt` file one line at a time. After each line this prompt appears:

```
Return to this state (y|n|Q)? y
```

Typing '`y`' at this point causes the independent plot variable(s) and all members of the guess vector to be returned to those values displayed in the file. Typing '`n`' (or simply <CR>) causes the next line to be displayed. '`Q`' skips to the end of the file

and makes no changes. No outputs are changed when this option is executed, so the model must be `(r)un` again to update them; however, be sure to disable the outer plot loop first if you want only one point. Alternatively, you can change the step or endpoints of the plot loop and start plotting again.

This option only works on the current (open) plot file, and it is not useful until after a run which has produced plot points.

`(E)xtras` The following model editing features are found under the `(E)xtras` submenu. Some less commonly used options (described in the next chapter) are also in this menu (v1.x DeltaE users will note that these options used to reside in the main menu):

`(S)plit segment.` This option automates the laborious process of splitting a duct segment (or anything else that has a length) into two segments each with half the original length, correcting the `sameas` and free target references, and correcting the iteration, optimization, and plot vectors. To partition the lengths differently, it is convenient to use `(s)pecial modes editing` to link the first length to the second, then `(m)odify` the first length, then clear (zero) the parameter linking before using the length in the iteration or optimization vector, if that is the intention. (All free targets, vectors, or `sameas` references to the segment specified are incremented by one; that is, the number of the original segment is incremented by one, and the 'clone' segment is effectively inserted before it.)

`(K)ill segment.` This option simply removes a segment from your model. Unlike `(S)plit segment`, however, it works on any type of segment (except `BEGIN`), and it does nothing intelligent with any lengths that are removed. The user must compensate another length where appropriate.

`(I)nsert segment.` DeltaE will prompt you for the correct number of parameters, giving the parameter name and units. This function is not perfectly interactive. If you make errors in typing in new parameter values, you will be left with a segment that is partly the same as the previous occupant of this spot. You may be able to recover by using the `(m)odify value` option in the main menu for numerical parameters. In the worst case (a bad segment type, for example), you may have to `(K)ill` the new segment and start over again. `(I)nsert` before #segments+1 is permitted to add a segment at the very end.

`(F)lip model.` For the same reason that DeltaE is most useful in the first place, that is, because an adequate set of boundary conditions is almost never known at the most convenient point to start calculations, the number of guesses and targets can sometimes be reduced by starting the integration of a model from what you previously considered the 'bottom.' Orifice pulse tube refrigerators (described in Chapter IV), are a particularly good example because they 'end' with a known impedance, but the 'beginning' driving impedance is generally unknown. The `(F)lip model` operation automates switching back and forth

between these two approaches to a solution, sparing the user from an effort that is otherwise tedious and very error prone. `(F)lip` reverses the order of every segment between the `BEGIN` and the last `HARDEND` or `SOFTEnd` and reverse their order. Segments within `TBRAN`ches are left in their original order, however. `sameas`, freetarget and plot references are all adjusted and an attempt is made to reform the guess and target vectors. Each `HXFRst` segment becomes an `HXLAst`, and *vice versa*.

Additional options are described at the end of Chapter V.

Chapter IV

Stirling Systems

Rott's equations implemented in DeltaE are valid for any phase difference between oscillatory pressure and oscillatory velocity, and any degree of thermal contact in the "stack". Hence, DeltaE can be used to model Stirling thermodynamic systems, in which p_1 and U_1 are substantially in phase, as well as thermoacoustic devices in which the phases p_1 and U_1 differ by nearly 90°. The principle additional DeltaE segment needed is one for stacked screen beds, because stacked screen regenerators are more common than parallel-plate, circular, or rectangular pore regenerators. In our opinion, the principle shortcomings of DeltaE for Stirling applications are DeltaE's acoustic approximation (which leads to reduced accuracy at high pressure amplitudes) and its inability to predict end effects and streaming-driven convective heat transport in pulse tubes (a shortcoming shared by many other design programs). Its principle virtues are speed and easy integral modeling of some auxiliary components such as ducts, dead volumes, and linear motors.

Harmonic analysis of Stirling systems is discussed by I. Urieli and D. M. Berchowitz, "Stirling Cycle Engine Analysis" (Hilger, Bristol, 1984) and by A. J. Organ, "Thermodynamics and Gas Dynamics of the Stirling Cycle Machine" (Cambridge University Press, 1992).

A Principles of Computation—Stacked Screens

The full details of the stacked-screen computation method implemented in DeltaE are described by G. W. Swift and W. C. Ward, "Simple harmonic analysis of stacked-screen regenerators," *J. Thermophys. and Heat Trans.* **10**, 652-662 (1996). As usual in DeltaE we adopt the point of view described at the beginning of Chapter III: We will regard p_1, U_1, and T_m as the dependent variables of interest. Given their values at one end, we can

generate $p_1(x)$, $U_1(x)$, and $T_m(x)$ throughout the regenerator, using equations of the form

$$dp_1/dx = F_1(p_1, U_1, T_m, \overline{H_2},\ \text{geometry}), \tag{IV.1}$$

$$dU_1/dx = F_2(p_1, U_1, T_m, \overline{H_2},\ \text{geometry}), \tag{IV.2}$$

$$dT_m/dx = F_3(p_1, U_1, T_m, \overline{H_2},\ \text{geometry}). \tag{IV.3}$$

The exact forms of these equations are displayed in Chapter VI below. Because p_1 and U_1 are complex, Eqs. (IV.1)-(IV.3) actually represent 5 real first-order differential equations. Equation (IV.1) is based largely on the screen friction factor data of Kays and London. Equation (IV.2) is based on the continuity equation, and Eq. (IV.3) on the equation for time-averaged energy flux $\overline{H_2}$ through the regenerator; both of the latter use the screen heat transfer coefficient data from Kays and London. The equations are not accurate for hydraulic radius on the order of δ_κ or greater.

The segment type implementing this algorithm is called `STKSCreen`. Corresponding heat exchangers, comprising stacked screens, are called `SXFRSt`, `SXMIDl`, and `SXLASt`, in which p_1 and U_1 are computed using Eqs. (IV.1) and (IV.2), with $dT_m/dx = 0$. As with the parallel-plate heat exchange segments `HX....`, a gas-to-metal temperature difference, proportional to the heat exchanger's heat flow, is also incorporated.

B Stirling Cryocooler

The sample files `Stirling.*` represent a simple 55 Hz, 2 MPa helium Stirling cryocooler with stacked-screen regenerator and heat exchangers. This apparatus is illustrated in Fig. IV.1. First, we examine `Stirling.out`:

```
TITLE     Bare bones Stirling cryocooler
!--------------------------------  0 --------------------------------
 BEGIN      Initialize things
  2.0000E+06 a Mean P    Pa               300.1     A T-beg  G( 0c)     P
   55.00     b Freq.     Hz              2.8520E+05 B |p|@0  G( 0d)     P
   300.1     c T-beg     K      G        -42.95     C Ph(p)0 G( 0e)     P
  2.8520E+05 d |p|@0     Pa     G        -36.02     D HeatIn G( 1e)     P
  -42.95     e Ph(p)0    deg    G
  3.6500E-04 f |U|@0    m^3/s
  0.0000     g Ph(U)0    deg
 helium     Gas type
 ideal      Solid type
!--------------------------------  1 --------------------------------
 SXFRST     hot heat exchanger
 sameas  2a  a Area      m^2             2.8145E+05 A |p|       Pa
  0.6000     b VolPor                    -43.66     B Ph(p)     deg
  1.0000E-03 c Length     m              3.6265E-04 C |U|     m^3/s
 sameas  2d  d   r_H      m             -0.3920     D Ph(U)     deg
```

```
 -36.02     e HeatIn     W     G         2.076     E Hdot      W
  300.0     f Est-T      K    = 1H?      37.16     F Work      W
sameas  0  Gas type                    -36.02     G Heat       W
copper     Solid type                   300.0     H MetalT     K
!--------------------------------  2 --------------------------------
STKSC      regenerator
 1.1670E-04 a Area      m^2            2.2874E+05 A |p|       Pa
 0.6860     b VolPor                   -52.95     B Ph(p)     deg
 5.0000E-02 c Length     m             6.2215E-05 C |U|     m^3/s
 1.3900E-05 d   r_H      m             -49.48     D Ph(U)     deg
 0.3000     e KsFrac                    2.076     E Hdot      W
                                        7.102     F Work      W
                                        300.1     G T-beg      K
sameas  0  Gas type                    79.96     H T-end      K
stainless  Solid type                  -30.06     I StkWrk     W
!--------------------------------  3 --------------------------------
SXLASt     cold heat exch
sameas  2a  a Area      m^2            2.2833E+05 A |p|       Pa
 0.6000     b VolPor                   -52.96     B Ph(p)     deg
 1.0000E-03 c Length     m             6.2000E-05 C |U|     m^3/s
sameas  2d  d   r_H      m             -52.00     D Ph(U)     deg
 0.0000     e HeatIn     W     (t)       7.077     E Hdot      W
  80.00     f Est-T      K    = 3H?      7.077     F Work      W
sameas  0  Gas type                     5.001     G Heat       W
copper     Solid type                   80.00     H MetalT     K
!--------------------------------  4 --------------------------------
FREETARG   U sub 1 at cold end
 6.2000E-05 a Target          = 4A?     6.2000E-05 A FreeT
 3C b ResAdr
!--------------------------------  5 --------------------------------
FREETARG   phase(U) at cold end
 -52.00     a Target          = 5A?     -52.00     A FreeT
 3D b ResAdr
!--------------------------------  6 --------------------------------
SOFTEnd    useless but required
 0.0000     a Re(Z)            (t)     2.2833E+05 A |p|       Pa
 0.0000     b Im(Z)            (t)     -52.96     B Ph(p)     deg
                                       6.2000E-05 C |U|     m^3/s
                                       -52.00     D Ph(U)     deg
                                        7.077     E Hdot      W
                                        7.077     F Work      W
                                        67.82     G Re(Z)
helium     Gas type                    -1.134     H Im(Z)
ideal      Solid type                   79.96     I   T       K

! The restart information below was generated by a previous run
! You may wish to delete this information before starting a run
! where you will (interactively) specify a different iteration
! mode.  Edit this table only if you really know your model!
INVARS      4  0  3  0  4  0  5  1  5
TARGS       4  1  6  3  6  4  1  5  1
SPECIALS    0
```

The real segments consist of a first heat exchanger, at 300 K, the regenerator, and a last heat exchanger at 80 K. All three are stacked screens. The other segments—`BEGIN`, the `FREETARGET`s, and `SOFTEND`—simply define the boundary conditions.

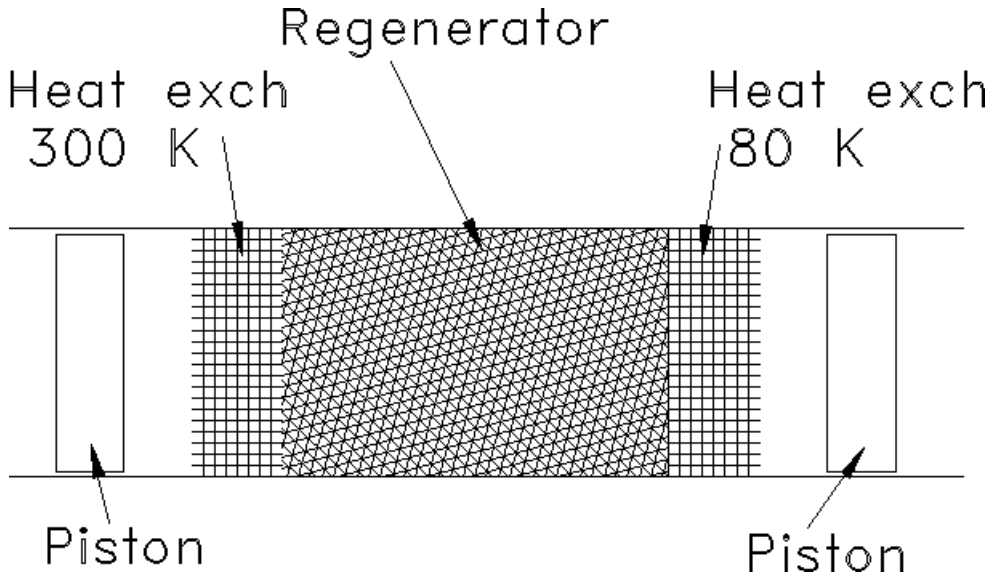


Figure IV.1: The Stirling cryocooler.

We arrived at the hydraulic radius r_h and volumetric porosity ϕ for the screens by hand, using expressions from Organ's book:

$$\phi = 1 - \frac{\pi m d}{4}\sqrt{1 + (md)^2}$$

$$r_h = \frac{d}{4}\frac{\phi}{1-\phi}$$

where d is wire diameter and m is mesh number (i.e., number of wires per unit length). The regenerator is a little over 1 cm in diameter and is 5 cm long. The heat exchangers are the same diameter but only 1 mm long.

DELTAE estimates the temperature difference between the helium gas and the copper screen wires in the heat exchangers, but it has no provision for estimating the temperature difference between the screen wires and the "housing" in which they are mounted (due to the finite thermal conductance of the screen wires themselves). This is not a serious concern for small machines, but should be checked by hand on a case-by-case basis.

Line e in the regenerator segment, "Ksfrac", is the fudge factor by which longitudinal conduction through the regenerator is reduced due to the spatially intermittent thermal contact between adjacent screens. Following Radebaugh, we often set Ksfrac=0.3 (*N.B:* rumor has it that he will announce a lower preferred value soon).

Our point of view with respect to boundary conditions in this example is most easily displayed by running DeltaE on this file and examining the vector summary

```
Iteration Vectors Summary:
 GUESS      0c          0d          0e          1e
 name  BEGIN:T-beg BEGIN:|p|@0 BEGIN:Ph(p) SXFRS:HeatI
 units      K           Pa          deg          W
 value    3.00E+02    2.93E+05    -43.        -37.
TARGET      1f          3f          4a          5a
 name  SXFRS:Est-T SXLAS:Est-T FREET:Targe FREET:Targe
 units      K           K
 value    3.00E+02     80.        6.20E-05    -52.
result      .00         .00         .00         .00
Potential TARGETS still available:
Addr Seg:Par-Type    Current Value
 3e SXLASt:HeatIn=    .0000         W
 6a SOFTEn:Re(Z) =    .0000
 6b SOFTEn:Im(Z) =    .0000
```

and the `BEGIN` segment above. Here, we are considering the volumetric velocities (both magnitudes and phases) at the two ends to be given, as if we have in mind an "alpha" Stirling machine, with two pistons determining the volumes of the compression and expansion spaces, respectively. The volumetric velocity at the hot end is set by lines f and g in the `BEGIN` segment. The 0° phase of line `0g` essentially determines the zero of phase for the entire system. The volumetric velocity 3.65×10^{-4} m^3/s of line `0f`, (together with the frequency set in line `0b`), implies a volumetric stroke of 2.1 cm^3 peak-to-peak at the hot end. The `FREETARGET`s at the cold end ensure that DeltaE's shooting method arrives there with the desired cold piston stroke and phase. To arrive at these two targets, DeltaE adjusts two guesses: the pressure amplitude and phase in the `BEGIN` segment (and hence throughout the cooler). We also insist that the metal temperatures in the two heat exchangers be 300 K and 80 K; DeltaE achieves these two targets by adjusting two more guesses: the heat extracted at the hot heat exchanger, and the temperature in the `BEGIN` segment.

DeltaE predicts that, under these circumstances, the cooler will reject 36 W at the hot heat exchanger and will have a cooling power of 5 W. This cooling power accounts for heat conduction and enthalpy flow through the regenerator, but does not account for any heat load imposed by the regenerator *case* conduction nor any load from frictional irreversibilities in the cold piston.

We now make or suggest a few simple modifications to this file to illustrate additional features of DeltaE.

To discover what temperature the cooler would maintain with a heat load of 10 W instead of 5 W, we (c)lear 3f—the cold heat exchanger temperature—from the target list. Instead, we (u)se 3e—the cooling power—as a target, and (m)odify it to 10 W. Running

DeltaE shows that under these circumstances the cold temperature will be 232 K. Using 3e as an independent (p)lot variable running from 10 W to 2 W with steps of, say, 0.5 W, and using 3H (cold metal temperature) as dependent (p)lot variable will generate a table of cold temperature (and other defaults) vs heat load:

gross cooling power SXLAS:HeatI	BEGIN:T-beg	BEGIN:|p|@0	BEGIN:Ph(p)	SXFRS:HeatI	metal temp @ cold hx SXLAS:Metal
W	K	Pa	deg	W	K
3e	0A	0B	0C	0D	3H
10.00	300.1	4.3154E+05	-71.57	-24.00	232.0
9.500	300.1	4.1420E+05	-70.09	-24.73	212.4
9.000	300.1	3.9704E+05	-68.40	-25.56	193.6
8.500	300.1	3.8016E+05	-66.45	-26.48	175.7
8.000	300.1	3.6368E+05	-64.22	-27.51	158.8
7.500	300.1	3.4778E+05	-61.67	-28.66	142.9
7.000	300.1	3.3265E+05	-58.75	-29.91	128.1
6.500	300.1	3.1853E+05	-55.43	-31.28	114.4
6.000	300.1	3.0571E+05	-51.69	-32.76	101.9
5.500	300.1	2.9449E+05	-47.52	-34.35	90.41
5.000	300.1	2.8518E+05	-42.94	-36.03	79.98
4.500	300.2	2.7809E+05	-37.99	-37.79	70.53
4.000	300.2	2.7348E+05	-32.74	-39.63	61.99
3.500	300.2	2.7154E+05	-27.31	-41.54	54.28
3.000	300.2	2.7240E+05	-21.81	-43.51	47.32
2.500	300.2	2.7609E+05	-16.38	-45.54	41.02
2.000	300.2	2.8252E+05	-11.14	-47.62	35.32

Insertion of two `ISPEAKer` segments before the aftercooler and after the cold heat exchanger would model use of linear motors driving pistons there.

Finally, in the next chapter "Advanced Features" we will use `TBRANCH` and `UNION` to change this model from an "alpha" Stirling machine to a "beta" or "gamma", with one power piston on the hot end and a displacer piston in parallel with the heat exchange elements.

C Pulse Tube Refrigerator

Changing a Stirling cryocooler into an orifice pulse tube refrigerator is a simple matter of replacing the cold piston with a pulse tube, heat exchanger, orifice, and reservoir volume in series. Fig. IV.2 represents such a cooler. The sample files `optr.*` represent a 300 Hz, 3 MPa helium orifice pulse tube refrigerator. After running DeltaE on `optr.in` or `optr.out`, we find the following `.out` and `.dat` files:

```
TITLE     an early Tektronix cooler design
```

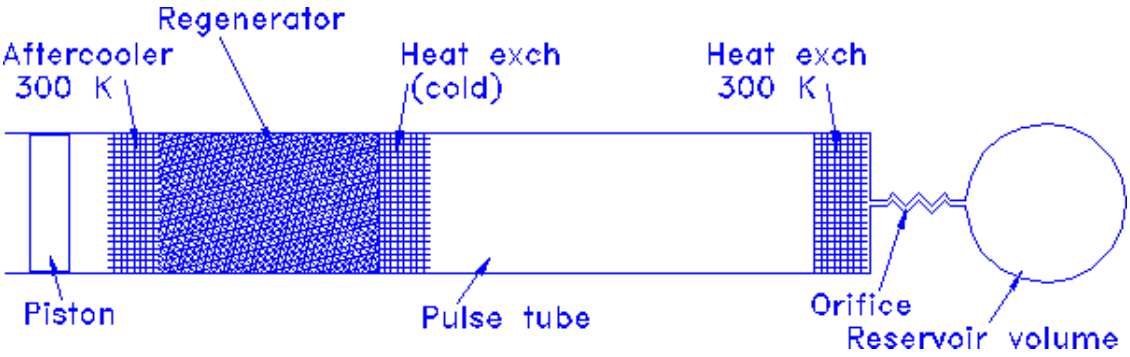


Figure IV.2: An Orifice Pulse Tube Refrigerator (OPTR).

```
!--------------------------------  0 --------------------------------
 BEGIN      Start with 8% p osc
  3.0000E+06 a Mean P    Pa              300.1     A T-beg  G( 0c)     P
   300.0     b Freq.     Hz             7.1473E-03 B |U|@0  G( 0f)     P
   300.1     c T-beg     K      G        50.44     C Ph(U)0 G( 0g)     P
  2.4000E+05 d |p|@0     Pa            -491.6     D HeatIn G( 1e)     P
  0.0000     e Ph(p)0    deg             5.338     E HeatIn G( 3e)     P
  7.1473E-03 f |U|@0    m^3/s   G
   50.44     g Ph(U)0    deg    G
 helium     Gas type
 ideal      Solid type
!--------------------------------  1 --------------------------------
 SXFRST     Aftercooler
  1.0290E-03 a Area      m^2             2.2896E+05 A |p|       Pa
  0.6900     b VolPor                    -3.587     B Ph(p)     deg
  1.2500E-02 c Length     m              6.0488E-03 C |U|     m^3/s
  6.4500E-05 d   r_H      m               44.21     D Ph(U)     deg
  -491.6     e HeatIn     W     G         54.65     E Hdot      W
   300.0     f Est-T      K    = 1H?      465.2     F Work      W
 helium     Gas type                    -491.6     G Heat       W
 copper     Solid type                   300.0     H MetalT     K
!--------------------------------  2 --------------------------------
 STKSCRN    Regenerator
 sameas  1a  a Area      m^2             1.6392E+05 A |p|       Pa
  0.7300     b VolPor                    -21.11     B Ph(p)     deg
  5.5000E-02 c Length     m              1.3354E-03 C |U|     m^3/s
  2.4000E-05 d   r_H      m              -24.44     D Ph(U)     deg
  0.3000     e KsFrac                     54.65     E Hdot      W
                                          109.3     F Work      W
                                          300.1     G T-beg      K
 helium     Gas type                     150.0     H T-end      K
 stainless  Solid type                  -355.9     I StkWrk     W
!--------------------------------  3 --------------------------------
 SXMIDL     Cold heat exchanger
 sameas  4a  a Area      m^2             9.4427E+04 A |p|       Pa
  0.6900     b VolPor                    -19.40     B Ph(p)     deg
  2.0000E-03 c Length     m              1.3347E-03 C |U|     m^3/s
  6.4500E-05 d   r_H      m              -24.70     D Ph(U)     deg
   5.338     e HeatIn     W     G         59.99     E Hdot      W
   150.0     f Est-T      K    = 3H?      62.75     F Work      W
 helium     Gas type                     5.338     G Heat       W
 copper     Solid type                   150.0     H MetalT     K
```

```
!--------------------------------  4 --------------------------------
 STKDU      Pulse tube
  5.6870E-05 a Area      m^2             9.8905E+04 A |p|       Pa
  2.6740E-02 b Perim      m    (-2)      -55.22     B Ph(p)     deg
  0.2000     c Length     m              1.2939E-03 C |U|     m^3/s
  1.0000E-05 d WallA     m^2            -42.68     D Ph(U)     deg
                                         59.99     E Hdot      W
                                         62.46     F Work      W
                                         150.0     G T-beg      K
 helium     Gas type                     300.2     H T-end      K
 stainless  Solid type                  -0.2886    I StkWrk     W
!--------------------------------  5 --------------------------------
 SXLAST     Hot heat exchanger
 sameas  4a  a Area      m^2             2.6242E+04 A |p|       Pa
  0.6900     b VolPor                    -103.4     B Ph(p)     deg
  5.0000E-03 c Length     m              1.2905E-03 C |U|     m^3/s
  6.4500E-05 d   r_H      m             -42.97     D Ph(U)     deg
  -23.00     e HeatIn     W     (t)       8.360     E Hdot      W
   300.0     f Est-T      K    = 5H?      8.360     F Work      W
 helium     Gas type                    -51.63     G Heat       W
 copper     Solid type                   300.0     H MetalT     K
!--------------------------------  6 --------------------------------
 IMPEDANCE  The orifice
  1.0000E+07 a Re(Zs) Pa-s/m^3           2.2821E+04 A |p|       Pa
  0.0000     b Im(Zs) Pa-s/m^3           -132.8     B Ph(p)     deg
                                         1.2905E-03 C |U|     m^3/s
                                         -42.97     D Ph(U)     deg
                                         3.3426E-02 E Hdot      W
 sameas  0  Gas type                     3.3426E-02 F Work      W
 ideal      Solid type                   -8.327     G HeatIn    W
!--------------------------------  7 --------------------------------
 COMPLIANCE Reservoir volume
  1.2680E-02 a Area      m^2   (-5)      2.2821E+04 A |p|       Pa
  1.5000E-04 b Volum     m^3             -132.8     B Ph(p)     deg
                                         1.2556E-08 C |U|     m^3/s
                                         -57.78     D Ph(U)     deg
                                         3.6948E-05 E Hdot      W
 sameas  0  Gas type                     3.6948E-05 F Work      W
 ideal      Solid type                  -3.3389E-02 G HeatIn    W
!--------------------------------  8 --------------------------------
 HARDEND    The end
  0.0000     a R(1/Z)          = 8G?     2.2821E+04 A |p|       Pa
  0.0000     b I(1/Z)          = 8H?     -132.8     B Ph(p)     deg
                                         1.2556E-08 C |U|     m^3/s
                                         -57.78     D Ph(U)     deg
                                         3.6948E-05 E Hdot      W
                                         3.6948E-05 F Work      W
                                         5.4890E-08 G R(1/Z)
 helium     Gas type                    2.0563E-07 H I(1/Z)
 ideal      Solid type                   300.2     I   T       K

! The restart information below was generated by a previous run
! You may wish to delete this information before starting a run
! where you will (interactively) specify a different iteration
! mode.  Edit this table only if you really know your model!
INVARS      5  0  3  0  6  0  7  1  5  3  5
TARGS       5  1  6  3  6  5  6  8  1  8  2
SPECIALS    2  4 -2  7 -5
```

```
 -= an early Tektronix cooler design                                    =-
 frequency=     300.000Hz     mean pressure=   3.000E+06Pa

  T(K)          p(Pa)               U(m^3/s)             hdot(W)   wdot(W)
   300.1    240000.       0.0      0.00455  0.00551         546.24    546.24
!-----------------------------------  1 -----------------------------------
 SXFRST    Aftercooler
 Heat =    -491.587 (W)    metal temp=     300.000 Kelvin
   300.1    228514.  -14323.4      0.00434  0.00422          54.65    465.19
!-----------------------------------  2 -----------------------------------
 STKSCRN   Regenerator
   150.0    152921.  -59043.8      0.00122 -0.00055          54.65    109.27
!-----------------------------------  3 -----------------------------------
 SXMIDL    Cold heat exchanger
 Heat =       5.338 (W)    metal temp=     150.000 Kelvin
   150.0     89065.  -31366.9      0.00121 -0.00056          59.99     62.75
!-----------------------------------  4 -----------------------------------
 STKDU     Pulse tube
   300.2     56422.  -81232.3      0.00095 -0.00088          59.99     62.46
!-----------------------------------  5 -----------------------------------
 SXLAST    Hot heat exchanger
 Heat =     -51.632 (W)    metal temp=     299.999 Kelvin
   300.2     -6074.  -25529.7      0.00094 -0.00088           8.36      8.36
!-----------------------------------  6 -----------------------------------
 IMPEDANCE The orifice
 Imped. work (heat extracted)=    8.33     Watts
   300.2    -15517.  -16733.8      0.00094 -0.00088           0.03      0.03
!-----------------------------------  7 -----------------------------------
 COMPLIANCEReservoir volume
 Heat extracted:   3.339E-02 Watts
   300.2    -15517.  -16733.8      0.00000  0.00000           0.00      0.00
!-----------------------------------  8 -----------------------------------
 HARDEND   The end
 inverse impedance (rho a U/p A)=(   5.489E-08,   2.056E-07)
   300.2    -15517.  -16733.8      0.00000  0.00000           0.00      0.00
```

The Stirling part of the system is modeled as a stacked-screen regenerator `STKSCREEN` and two stacked-screen heat exchangers `SXFRST` and `SXMIDL`. We model the pulse tube itself as a `STKDUCT`, using Rott's wave equation and enthalpy flux equation in boundary-layer approximation, because the tube diameter is $\gg \delta_\kappa$. (We will discuss this approximation shortly.) The heat exchanger at the hot end of the pulse tube is the `HXLAST`. The orifice and reservoir volume are easily modeled as a DeltaE `IMPEDance` and `COMPLiance`, respectively. Our use of zero for the imaginary part of the `IMPEDance` reflects our intention that this orifice will truly be resistive, with pressure drop in phase with mass flux.

For purposes of illustration here, we will regard the geometry of the apparatus as given, and will explore its performance. The vector summary indicates our point of view:

```
Iteration Vectors Summary:
 GUESS      0c          0f          0g          1e          3e
 name  BEGIN:T-beg BEGIN:|U|@0 BEGIN:Ph(U) SXFRS:HeatI SXMID:HeatI
 units      K          m^3/s        deg          W           W
```

```
 value    3.00E+02    7.15E-03     50.        -4.92E+02     5.3
TARGET      1f          3f          5f          8a          8b
 name  SXFRS:Est-T SXMID:Est-T SXLAS:Est-T HARDE:R(1/Z HARDE:I(1/Z
 units      K           K           K
 value    3.00E+02    1.50E+02    3.00E+02     .00          .00
result    3.00E+02    1.50E+02    3.00E+02    7.61E-08   -3.47E-08
```

Three of the 5 targets fix the hot and cold temperatures at 300 K and 150 K. We leave the amplitude of the oscillatory pressure at the `BEGINning` at 8% of mean pressure, and leave the frequency fixed at 300 Hz. Hence, we are asking: What is the cooling power at 150 K, and how much input power, volumetric velocity, etc. are required, for fixed frequency and pressure amplitude? The result, given in the file listings above: 5.34 W of cooling power, requiring 546 W of input power from the compressor.

Our choice of $\mathrm{Re}(Z_s) = 1 \times 10^7$ for the orifice impedance above was random. To find a better orifice setting, we can use $\mathrm{Re}(Z_s)$ as an independent plot variable, letting it range from 1×10^7 to 1×10^8. The cooling power peaks at 7.58 W for $\mathrm{Re}(Z_s) = 4.7 \times 10^7$.

As with most optr models we have worked with in DeltaE this one is not particularly "robust". A change of a typical variable by 20% or 30% will likely cause DeltaE to get hopelessly lost. Hence the steps we used in the plotting of $\mathrm{Re}(Z_s)$ were small: 1×10^6. Part of the "'fragility" of optr models (as compared to thermoacoustic models) in DeltaE is due to the fact that small changes in variables near the `BEGINning`, such as p_1, U_1, and the heat removed at the aftercooler, have a large effect on temperatures at the end of the pulse tube. Some of the fragility is due to the fact that optr models typically have a large number of guesses and targets. Thus, when you encounter a fragile DeltaE model, try to reduce the number of guesses and targets as much as possible (particularly in initial design explorations when you are more lost than DeltaE) and, once you have a convergent model, make only small changes in variables. Tighten up DeltaE's convergence tolerance if you have to use more than 5 guesses and 5 targets. To accomplish a large change in a variable, use `(p)lot` to break it up into many small steps. A fast computer and frequent saving of satisfactory converged models will minimize frustration.

Examination of the pulse tube segment in the `.dat` file above shows a possible problem: The pulse tube figure of merit, which Radebaugh defines as $\dot{H}/\dot{W}$, is high: $\dot{H}/\dot{W} \simeq$ 60 W/63 W$\simeq$ 0.95. A more common experimental value of pulse tube figure of merit is 0.7. DeltaE knows nothing about jet-, turbulence-, or streaming-driven convection, and pulse-tube experimentalists have not yet learned how to reliably avoid such convection. For a discussion of streaming-driven convection, see "Acoustic streaming in pulse tube refrigerators: Tapered pulse tubes," J. R. Olson and G. W. Swift, to be published in Cryogenics, 1997.

To force DeltaE to accomodate such a reduced pulse tube figure of merit, you can introduce a new segment `QUOTARGET` (short for quotient target, discussed more fully in the next chapter) and an additional guess/target pair. The quotient target is used to maintain the quotient of any two output variables fixed. Hence, it can maintain the ratio of pulse tube enthalpy flux to pulse tube work flux equal to 0.7. You can simulate the thermal loading of streaming-driven convection, etc. by letting DeltaE guess an unphysically large value for the cross section of the pulse tube wall (line 4d), which then conducts significant heat from hot to cold, allowing DeltaE to meet its target of 0.7. We will not do so here.

We can improve the overall performance of this refrigerator by a simple means, similar in principle to the second orifice of a double-inlet pulse tube refrigerator: adding a small duct between the orifice and reservoir volume adds inertance to the impedance of the end of the system; proper choice of the length/area of this duct can phase shift the mass flow through the orifice significantly. This is entirely analogous to putting an inductor in series with an *RC* circuit, and is described in "Use of inertance in orifice pulse tube refrigerators," D. L. Gardner and G. W. Swift, to be published in Cryogenics in 1997, and references therein. Adding an inertance to our model, and adjusting its area/length and Re(Z_s) of the orifice for maximum cooling power brings the cooling power up to 11.7 W in the `.out` file below.

```
TITLE     an early Tektronix cooler design
!->optr2.out
!Created@11:46: 6 23-May-97 with DeltaE Vers. 3.5b1 for the IBM/PC-Compatible
!--------------------------------  0 ---------------------------------
 BEGIN      Start with 8% p osc
  3.0000E+06 a Mean P    Pa               300.1     A T-beg  G( 0c)     P
   300.0     b Freq.     Hz             7.1813E-03 B |U|@0  G( 0f)     P
   300.1     c T-beg     K      G        52.28     C Ph(U)0 G( 0g)     P
  2.4000E+05 d |p|@0     Pa              -473.8     D HeatIn G( 1e)     P
  0.0000E+00 e Ph(p)0    deg              11.75     E HeatIn G( 3e)     P
  7.1813E-03 f |U|@0    m^3/s   G
   52.28     g Ph(U)0    deg    G
 helium     Gas type
 ideal      Solid type
!--------------------------------  1 ---------------------------------
 SXFRST     Aftercooler
  1.0290E-03 a Area      m^2             2.2942E+05 A |p|       Pa
  0.6900     b VolPor                    -3.702     B Ph(p)     deg
  1.2500E-02 c Length     m              6.0603E-03 C |U|     m^3/s
  6.4500E-05 d   r_H      m               46.38     D Ph(U)     deg
  -473.8     e HeatIn     W     G         53.41     E Hdot      W
   300.0     f Est-T      K    = 1H?      446.1     F Work      W
 helium     Gas type                    -473.8     G Heat       W
 copper     Solid type                   300.0     H MetalT     K
!--------------------------------  2 ---------------------------------
 STKSCRN    Regenerator
 sameas  1a  a Area      m^2             1.6909E+05 A |p|       Pa
  0.7300     b VolPor                    -21.79     B Ph(p)     deg
  5.5000E-02 c Length     m              1.2017E-03 C |U|     m^3/s
  2.4000E-05 d   r_H      m              -23.78     D Ph(U)     deg
  0.3000     e KsFrac                     53.41     E Hdot      W
                                          101.5     F Work      W
```

```
                                        300.1     G T-beg      K
 helium     Gas type                    149.9     H T-end      K
 stainless  Solid type                 -344.6     I StkWrk     W
!--------------------------------  3 --------------------------------
 SXMIDL     Cold heat exchanger
 sameas  4a  a Area      m^2           1.1266E+05 A |p|        Pa
  0.6900     b VolPor                  -21.34     B Ph(p)      deg
  2.0000E-03 c Length     m            1.2007E-03 C |U|      m^3/s
  6.4500E-05 d   r_H      m            -24.10     D Ph(U)      deg
   11.75     e HeatIn     W     G       65.16     E Hdot       W
   150.0     f Est-T      K    = 3H?    67.56     F Work       W
 helium     Gas type                    11.75     G Heat       W
 copper     Solid type                  150.0     H MetalT     K
!--------------------------------  4 --------------------------------
 STKDU      Pulse tube
  5.6870E-05 a Area      m^2           1.1252E+05 A |p|        Pa
  2.6740E-02 b Perim      m    (-2)     -48.69     B Ph(p)      deg
  0.2000     c Length     m            1.1960E-03 C |U|      m^3/s
  1.0000E-05 d WallA      m^2          -47.60     D Ph(U)      deg
                                        65.16     E Hdot       W
                                        67.28     F Work       W
                                        149.9     G T-beg      K
 helium     Gas type                   300.1     H T-end      K
 stainless  Solid type                -0.2844     I StkWrk     W
!--------------------------------  5 --------------------------------
 SXLAST     Hot heat exchanger
 sameas  4a  a Area      m^2           4.0780E+04 A |p|        Pa
  0.6900     b VolPor                  -52.34     B Ph(p)      deg
  5.0000E-03 c Length     m            1.1947E-03 C |U|      m^3/s
  6.4500E-05 d   r_H      m            -48.04     D Ph(U)      deg
  -23.00     e HeatIn     W     (t)      24.29     E Hdot       W
   300.0     f Est-T      K    = 5H?     24.29     F Work       W
 helium     Gas type                   -40.87     G Heat       W
 copper     Solid type                  300.0     H MetalT     K
!--------------------------------  6 --------------------------------
 IMPEDANCE  The orifice
  3.4037E+07 a Re(Zs) Pa-s/m^3          3054.     A |p|        Pa
  0.0000E+00 b Im(Zs) Pa-s/m^3         -138.0     B Ph(p)      deg
                                       1.1947E-03 C |U|      m^3/s
                                       -48.04     D Ph(U)      deg
                                       1.4901E-03 E Hdot       W
 sameas  0  Gas type                   1.4901E-03 F Work       W
 ideal      Solid type                 -24.29     G HeatIn     W
!--------------------------------  7 --------------------------------
 ISODUCT    inertance
  2.8000E-02 a Area      m^2            3061.     A |p|        Pa
  0.5932     b Perim      m    (-2)     -138.0     B Ph(p)      deg
  3.1620E-02 c Length     m            1.7312E-04 C |U|      m^3/s
  3.0000E-04 d Srough                  -48.13     D Ph(U)      deg
                                       6.0081E-04 E Hdot       W
 sameas  0  Gas type                   6.0081E-04 F Work       W
 ideal      Solid type                -8.8931E-04 G HeatIn     W
!--------------------------------  8 --------------------------------
 COMPLIANCE Reservoir volume
  1.2680E-02 a Area      m^2   (-5)     3061.     A |p|        Pa
  1.5000E-04 b Volum     m^3           -138.0     B Ph(p)      deg
                                       4.6748E-11 C |U|      m^3/s
                                        15.51     D Ph(U)      deg
```

```
                                        -6.4043E-08 E Hdot      W
 sameas  0  Gas type                    -6.4043E-08 F Work      W
 ideal      Solid type                  -6.0087E-04 G HeatIn    W
!--------------------------------  9 ---------------------------------
 HARDEND    The end
  0.0000E+00 a R(1/Z)          = 9G?      3061.     A |p|       Pa
  0.0000E+00 b I(1/Z)          = 9H?     -138.0     B Ph(p)     deg
                                         4.6748E-11 C |U|     m^3/s
                                          15.51     D Ph(U)     deg
                                        -6.4043E-08 E Hdot      W
                                        -6.4043E-08 F Work      W
                                        -5.2868E-09 G R(1/Z)
 helium     Gas type                    2.6354E-09 H I(1/Z)
 ideal      Solid type                   300.1     I   T       K

! The restart information below was generated by a previous run
! You may wish to delete this information before starting a run
! where you will (interactively) specify a different iteration
! mode.  Edit this table only if you really know your model!
INVARS      5  0  3  0  6  0  7  1  5  3  5
TARGS       5  1  6  3  6  5  6  9  1  9  2
SPECIALS    3  4 -2  7 -2  8 -5
```

Tektronix researchers used DeltaE to model a 350 Hz orifice pulse tube refrigerator, as described in the article by Godshalk *et al.* in the proceedings of the 1995 Cryogenic Engineering Conference: *Advances in Cryogenic Engineering*,**41**, 1411-1418 (Plenum, New York, 1996).

D Etched Foil Regenerators

The segment `STKPOwerlaw` is intended to model regenerators for which the friction factor and heat transfer coefficient are power laws in Reynold's number. This include Ran Yaron's etched foil regenerators. For input syntax and mathematical details, see the end of section VI.B.5.

Chapter V

Advanced Features

This chapter introduces additional features of DeltaE that expand its uses and convenience for the user who is already comfortable with the basics. Here we explain "free targets" that allow the increased control over the endpoints of DeltaE's iterations and incorporate some basic math functions; active branches that permit simultaneous calculation of side branches and main ducts for complicated models; additional fluid options, including binary gas mixtures; parameter linking so that iterations can be performed while maintaining certain geometric relationships in the model; and several other useful features and tunable parameters.

A Free Targets

DeltaE reserves a place for a special input parameter to hold a target value in the segment types that have outputs commonly used in targets; these parameters are: heat exchanger temperatures and heat flows and complex impedances in `HARD-` and `SOFTEnd` segments (`UNION`s, introduced in the next section, are a special case). The code knows, internally, to pair these input values with the appropriate output results of the segment for comparison. The experienced user, however, will soon hunger for more possibilities once a model is defined and converging to meet the basic targets. An application may call for work, pressure, or velocity (magnitude or phase) to be specified at a certain point, or some derived function of outputs may be desired for targeting or plotting. Free targets are used for these purposes. We can also use them to generate a new type of output based on other outputs in the model; in this case, the first 'target' parameter is simply ignored. All free targets have one real input and one real output which DeltaE recognizes as a potential target/result pair. The other input parameters to these segments are one or more addresses.

There are 7 types of free targets: `FREETarget`, `QUOTArget`, `COPRTarget`, `EFFRTarget`, `PRODTarget`, `DIFFTarget`, and `VOLMTarget`. The latter six perform some basic math on the outputs they reference. For more complicated functions, free targets can be cascaded. However, it is not our intention to provide a complete (convoluted!) mathematical language using these modules. For more elaborate post processing of outputs, we encourage the use of spreadsheets or math-capable stream-edit tools such as `awk` or `perl`.

The "target" parameter of free targets, like that of any other targets, can be used as the independent variable in a plotting loop. All free targets should be placed after all the segments that they reference in the model so that, during processing (which is sequential), they will be updated with the most recent results. Free target specifications do *not* end with fluid and solid names like normal segments.

`FREETarget.` The basic form of free target simply allows you to specify one additional input value that you wish to have the solver compare with an output value that it normally does not consider. It has two input parameters: the real target value, and an output address (*e.g.* 5F).

`QUOTArget.` Almost the same as the `FREETarget`, the `QUOTArget` adds an additional output address; it generates its output value from the quotient of the first output (`NumAdr`) divided by the second output (`DenomAdr`). One obvious use for such a module is to generate an efficiency, W/Q, or a COP. The efficiency may be used as a target (which may or may not converge, depending on how greedy you are), or it can simply be inserted as a plot parameter, to save the trouble of doing the calculation later.

`COPRTarget, EFFRTarget.` These two *relative* targets are quotient targets taken a step further: they also use the addresses of two temperatures, `ThAdr` and `TcAdr`, so that DeltaE can take the quotient and normalize it by Carnot's COP or efficiency. The `COPRTarget` has already been introduced in the previous chapter, in one of the Hofler refrigerator examples. `COPRTarget` computes

$$\frac{Q_c}{W}\frac{T_h - T_c}{T_c};$$

`EFFRTarget` computes

$$\frac{W}{Q_h}\frac{T_h}{T_h - T_c}.$$

It is up to the user to ensure that the addresses given are truly what they are said to be. (Note: the `COPRTarget` and `EFFRTarget` can be set up to use gas temperatures, parameters G and H in the stack, instead of metal temperatures from the heat exchangers. The effect is to give system performance for ideal heat exchangers.)

`PRODTarget.` Similar to the quotient target, a product target takes two output addresses but generates its output from their product. This module can be useful for generating

some figure-of-merit that is to be maximized over a series of plot points. For example, COPR $*$ COPR $*$ Q_c can be generated from a `COPRTarget` and two `PRODTarget`s by the following model fragment:

```
!--------------------------------- 10 ---------------------------------
 COPRT      COP/COP-Carnot
   0.000     a Target           (t)       0.0130     A COP/Cc
 5G         NumAdr
 1F         DenomAdr
 3H         ThAdr
 5H         TcAdr
!--------------------------------- 11 ---------------------------------
 PRODT
   0.000     a Target           (t)       0.0169     A Prod.
 10A         b M1Addr
 10A         c M2Addr
!--------------------------------- 12 ---------------------------------
 PRODT
   0.000     a Target           (t)       8.45       A Prod.
 11A        b  M1Adr
 5G         c  M2Adr
```

(This is about as extreme an example of 'free target mathematics' as we would like to envision.)

`DIFFTarget.` The difference target, which has been introduced in the final Hofler refrigerator example to maintain resonance at the speaker, is like quotient and product targets in that it takes two output addresses; it generates an output equal to their difference. These targets are often used for phases, so differences of 0 or ± 90 will be common targets.

`VOLMTarget.` The volume target has two address parameters, and it simply generates an output equal to the total volume contained in segments between those two addresses (inclusive). Only the segment number is used; the parameter letter is ignored. The first segment number must be less than or equal to the second. This target is intended to give an indication of the overall size and weight of a design (or a portion of it) for doing tradeoff analysis.

`CONSTants:` While `CONSTants` is not itself a free target segment, its use is usually in conjunction with them. This segement simply copies input parameters to the output parameter space where they can be used in "free target mathematics." It also copies (and scales) the current plot loop values (which are inputs) into the output space since they would not otherwise be available.

For a concise listing of all the free targets and their parameters, please consult the reference section in Chapter VI.

B Active Branches

Although `BRANCh` and `OPNBRanch` have their uses, they are often inadequate for describing the variations in branch impedance with operating conditions. For example, the branch might be a Helmholtz resonator whose impedance changes significantly with frequency. Further, `BRANCh` and `OPNBRanch` are wholly inadequate when branches involve thermoacoustic components. The `TBRANch` segment addresses these inadequacies by allowing DeltaE to integrate its way down a side branch and then return to the trunk and integrate there as before.

As an example, consider the modification shown in Figure V.1 to the basic "beer cooler" (heat-driven thermoacoustic refrigerator) shown in Figure I.7. We might want to investigate whether performance would improve by adding the side branch so that the entire volume velocity required by the prime-mover stack would no longer have to flow through the refrigerator stack and much of the resonator dissipation would show up at ambient temperature instead of at the cold heat exchanger.

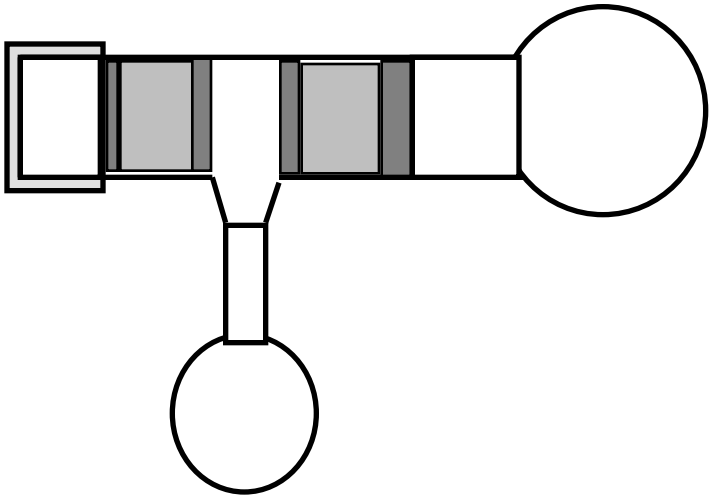


Figure V.1: Modified "beer cooler."

DeltaE uses the `TBRANch` segment for cases like this. When it encounters a `TBRANch`, DeltaE treats subsequent segments as the sequential members of the branch, until it reaches a `HARDEnd` or `SOFTEnd`. It then "returns to the trunk," treating the rest of the segments as trunk members. So the sequence of segments for the example of Figure IV.1 might be as follows:

```
TITLE
BEGIN      0
ENDCAP     1
```

```
ISODUCT    2
HXFRST     3
STKSLAB    4
HXLAST     5
TBRANCH    6
  ISOCONE    7
  ISODUCT    8
  COMPLIANCE 9
  HARDEND   10
HXFRST    11
STKSLAB   12
HXLAST    13
ISODUCT   14
COMPLIAN  15
HARDEND   16
```

Segments 5 through 9 comprise the side branch; the others comprise the trunk.

The method of computation is as follows. At a branch, the branch impedance determines how the (complex) volume velocity splits up at the branch. Often, we use the branch impedance as a pair of guesses that DeltaE adjusts in its usual way to get the complex impedance at the next 'END to come out right. (If asked to do so, DeltaE should pair select both of these guess and target pairs as part of a default set. If not, you should enable them.) TBRANCHed models tend to have guess and target vectors of high dimension, since every 'END contributes two targets (and a few more targets are almost always needed for temperatures, heats, etc.). Stacks and heat exchangers can also be used in branches, and, of course, branches can have subbranches of their own.

TBRANch has a companion segment type, TEE, that takes the filename of another valid DeltaE input file as its only parameter. When DeltaE encounters a TEE, it loads the named file into the model, and replaces the BEGIN segment of the branch file with a TBRANch segment. It tries to guess starting values for the complex branch impedance, and then adjusts the addresses in any sameas declarations and free target-type segments occurring in the branch (or after the branch point) by the number of segments in the branch. Once the file has been read in, the TEE segment disappears—the .out file and (d)isplayed segments will be the composite model.

When rewriting our previous example to use a TEE segment, the model has the form

```
TITLE
BEGIN      0
ENDCAP     1
ISODUCT    2
HXFRST     3
STKSLAB    4
HXLAST     5
TEE        6
 branch.in
```

```
HXFRST     7
STKSLAB    8
HXLAST     9
ISODUCT   10
COMPLIAN  11
HARDEND   12
```

where we have omitted the parameters of all but the `TEE` segment. The file `branch.in` is a valid DeltaE input file, which we have run and debugged separately. This input file looks like this:

```
TITLE
BEGIN      0
ISOCONE    1
ISODUCT    2
COMPLIANCE 3
HARDEND    4
```

The file may have any name (*e.g.*, `branch.in, branch.out, branch.tee`), but it must be specified with the complete suffix.

The two models above will combine to produce the same model as our first example. This approach is recommended, especially for nontrivial branches containing stacks, etc., so that the two simpler submodels can be evaluated first. The impedance that DeltaE chooses for the `TBRANch` may need immediate attention; guess and target vectors, free targets, and `sameas` references should also be checked carefully. Special modes (see below) that link length parameters across the branch point will not be handled properly, and must be redone with new segment numbers.

The multiply-connected duct network of Figure I.3 can also be handled by DeltaE, through use of `TBRANch` and `UNION`. The `UNION` segment is used to tell DeltaE to "connect" a `TBRANch`'s `SOFTEnd` (or `HARDEnd`) back to the trunk at the location of the `UNION` segment. The branch's' `SOFTEnd` impedance targets are no longer used; instead, the two real input variables (`b` and `c`) of the `UNION` segment should always be active targets. It does not matter what the initial values of these parameters are; as soon as DeltaE processes the segment, it copies in the current values of the complex pressure at the `SOFTEnd` referenced by the number in parameter `a` of the `UNION` segment. These values are compared to the local complex pressure result, at this `UNION`, in the trunk, and iteration should drive the model until their difference is zero. In other words, when a branch and trunk meet at a `UNION`, they must share the same complex p_1. As before, a guessed branch impedance usually determines how the (complex) volume velocity splits up at the `TBRANch`. Volume velocities are summed at the `UNION`. (The `UNION` segment is somewhat similar to the freetargets of the previous section, except that it grabs two results simultaneously, from fixed locations

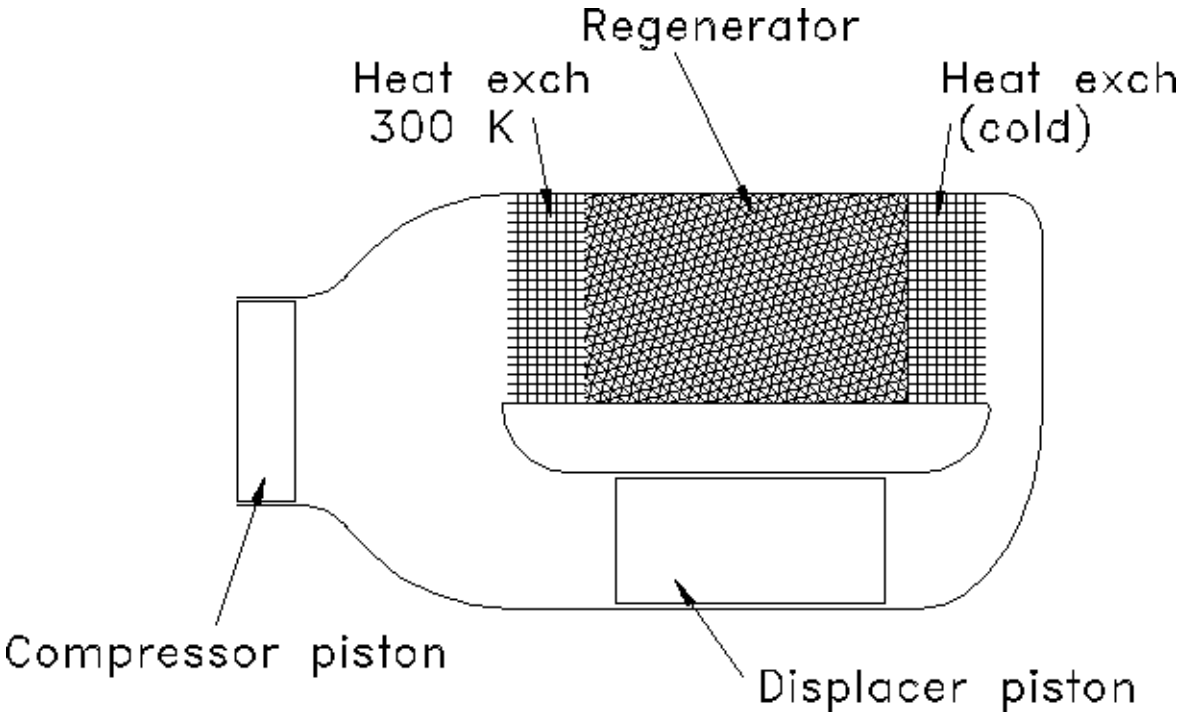


Figure V.2: "Gamma"-style Stirling machine.

within the referenced segment. Also, the 'target' values are not specified by the user, since they move dynamically depending on what is happening at the attached '`End` segment.)

As an example of use of `TBRANch` and `UNION`, we return to the Stirling cryocooler example of the previous chapter, and convert it to a "gamma" style Stirling machine, with a compressor piston at the hot end and a displacer piston connecting the hot and cold ends. In the previous example, *PU* power flowed in at the `BEGIN` and out at the `...END`; with a displacer piston, the cold-end *PU* power is returned automatically to the hot end, reducing the hot-end *PU* power requirement.

The apparatus layout is illustrated in Fig. V.2; the corresponding DeltaE file layout is

```
TITLE
BEGIN
   TBRANCH
   IMPEDANCE (the displacer)
   SOFTEND
SXFRST
STKSCREEN
SXLAST
UNION  ('connects' to softend above)
HARDEND
```

and the corresponding `.out` file is

```
TITLE     Stirling cooler w displacer piston, illustrating TBRANCH--UNION
!--------------------------------  0 --------------------------------
 BEGIN      Initialize things
  2.0000E+06 a Mean P    Pa              300.1     A T-beg  G( 0c)     P
   55.00     b Freq.     Hz             2.8597E+05 B |p|@0  G( 0d)     P
   300.1     c T-beg     K      G        -42.49     C Ph(p)0 G( 0e)     P
  2.8597E+05 d |p|@0     Pa     G       -4.5489E+09 D Re(Zb) G( 1a)     P
  -42.49     e Ph(p)0    deg    G       -7.6222E+08 E Im(Zb) G( 1b)     P
  3.3000E-04 f |U|@0    m^3/s           -8.5227E+08 F Re(Zs) G( 2a)     P
   9.000     g Ph(U)0    deg            -7.8741E+08 G Im(Zs) G( 2b)     P
 helium     Gas type                    -36.05     H HeatIn G( 4e)     P
 ideal      Solid type
!--------------------------------  1 --------------------------------
 TBRANCH    branch to displacer
 -4.5489E+09 a Re(Zb) Pa-s/m^3  G       2.8597E+05 A |p|       Pa
 -7.6222E+08 b Im(Zb) Pa-s/m^3  G       -42.49     B Ph(p)     deg
                                        6.2000E-05 C |U|     m^3/s
                                         128.0     D Ph(U)     deg
                                        -8.743     E Hdot      W
 sameas  0  Gas type                    -8.743     F Work      W
 ideal      Solid type                   38.12     G Work_T    W
!--------------------------------  2 --------------------------------
 IMPEDANCE  displacer, sort of
 -8.5227E+08 a Re(Zs) Pa-s/m^3  G       2.2920E+05 A |p|       Pa
 -7.8741E+08 b Im(Zs) Pa-s/m^3  G       -52.39     B Ph(p)     deg
                                        6.2000E-05 C |U|     m^3/s
                                         128.0     D Ph(U)     deg
                                        -7.105     E Hdot      W
 sameas  0  Gas type                    -7.105     F Work      W
 ideal      Solid type                   1.638     G HeatIn    W
!--------------------------------  3 --------------------------------
 SOFTEND    reconnect at UNION
  0.0000     a Re(Z)            (t)     2.2920E+05 A |p|       Pa
  0.0000     b Im(Z)            (t)     -52.39     B Ph(p)     deg
                                        6.2000E-05 C |U|     m^3/s
                                         128.0     D Ph(U)     deg
                                        -7.105     E Hdot      W
                                        -7.105     F Work      W
                                        -131.9     G Re(Z)
 sameas  0  Gas type                    0.8990     H Im(Z)
 ideal      Solid type                   300.1     I   T       K
!--------------------------------  4 --------------------------------
 SXFRST     Hot heat exchanger
 sameas  5a  a Area      m^2            2.8223E+05 A |p|       Pa
  0.6000     b VolPor                   -43.20     B Ph(p)     deg
  1.0000E-03 c Length     m             3.6176E-04 C |U|     m^3/s
  sameas  5d d   r_H      m             4.1153E-02 D Ph(U)     deg
  -36.05     e HeatIn     W     G        2.072     E Hdot      W
   300.0     f Est-T      K    = 4H?     37.19     F Work      W
 sameas  0  Gas type                    -36.05     G Heat       W
 copper     Solid type                   300.0     H MetalT     K
!--------------------------------  5 --------------------------------
 STKSC      regenerator
  1.1670E-04 a Area      m^2            2.2961E+05 A |p|       Pa
  0.6860     b VolPor                   -52.39     B Ph(p)     deg
```

```
 5.0000E-02 c Length     m              6.2189E-05 C |U|     m^3/s
 1.3900E-05 d   r_H      m              -49.47     D Ph(U)     deg
 0.3000     e KsFrac                     2.072     E Hdot      W
                                         7.130     F Work      W
                                         300.1     G T-beg      K
 sameas  0  Gas type                     79.96     H T-end      K
 stainless  Solid type                  -30.06     I StkWrk     W
!--------------------------------  6 ---------------------------------
 SXLASt     cold heat exch
 sameas  5a  a Area      m^2            2.2920E+05 A |p|       Pa
  0.6000     b VolPor                   -52.39     B Ph(p)     deg
  1.0000E-03 c Length     m             6.2000E-05 C |U|     m^3/s
  sameas  5d d   r_H      m             -52.00     D Ph(U)     deg
  0.0000     e HeatIn     W     (t)       7.105     E Hdot      W
   80.00     f Est-T      K    = 6H?      7.105     F Work      W
 sameas  0  Gas type                     5.033     G Heat       W
 copper     Solid type                   80.00     H MetalT     K
!--------------------------------  7 ---------------------------------
 UNION      displacer cold end
   3.000     a TendSg                   2.2920E+05 A |p|       Pa
  2.2920E+05 b |p|End    Pa    = 7A?     -52.39     B Ph(p)     deg
  -52.39     c Ph(p)E    deg   = 7B?     4.4516E-11 C |U|     m^3/s
                                          88.75     D Ph(U)     deg
                                       -3.9725E-06 E Hdot      W
 sameas  0  Gas type                   -3.9725E-06 F Work      W
 ideal      Solid type                   0.0000     G HeatIn    W
!--------------------------------  8 ---------------------------------
 FREETARGET displacer U
  6.2000E-05 a Target          = 8A?     6.2000E-05 A FreeT
  6C b ResAdr
!--------------------------------  9 ---------------------------------
 FREETARGET displacer phase
  -52.00     a Target          = 9A?     -52.00     A FreeT
  6D b ResAdr
!-------------------------------- 10 ---------------------------------
 HARDEND    the end
  0.0000     a R(1/Z)          =10G?     2.2920E+05 A |p|       Pa
  0.0000     b I(1/Z)          =10H?     -52.39     B Ph(p)     deg
                                       4.4516E-11 C |U|     m^3/s
                                        88.75     D Ph(U)     deg
                                      -3.9725E-06 E Hdot      W
                                      -3.9725E-06 F Work      W
                                      -8.2110E-09 G R(1/Z)
 helium     Gas type                   6.6160E-09 H I(1/Z)
 ideal      Solid type                  79.96     I   T       K

! The restart information below was generated by a previous run
! You may wish to delete this information before starting a run
! where you will (interactively) specify a different iteration
! mode.  Edit this table only if you really know your model!
INVARS      8  0  3  0  4  0  5  1  1  1  2  2  1  2  2  4  5
TARGS       8  4  6  6  6  7  2  7  3  8  1  9  1 10  1 10  2
SPECIALS    0
```

We are using an `IMPEDance` segment as the displacer piston. The real part of its impedance tells us how much power is needed to move the displacer, in order to overcome

friction and viscous losses in the regenerator and heat exchangers. The imaginary part of the IMPEDance's impedance tells us whether the displacer needs to look massy (positive values) or springy (negative values) in order to move as desired.

Our guess/target vector summary is the largest we have yet encountered in this user's guide—eight each:

```
Iteration Vectors Summary:
 GUESS      0c          0d          0e          1a          1b          2a
 name  BEGIN:T-beg BEGIN:|p|@0 BEGIN:Ph(p) TBRAN:Re(Zb TBRAN:Im(Zb IMPED:Re(Zs
 units      K           Pa          deg      Pa-s/m^3    Pa-s/m^3    Pa-s/m^3
 value    3.00E+02    2.86E+05    -43.       -4.55E+09   -7.62E+08   -8.52E+08
 GUESS      2b          4e
 name  IMPED:Im(Zs SXFRS:HeatI
 units  Pa-s/m^3        W
 value   -7.87E+08    -36.
TARGET      4f          6f          7b          7c          8a          9a
 name  SXFRS:Est-T SXLAS:Est-T UNION:|p|En UNION:Ph(p) FREET:Targe FREET:Targe
 units      K           K          Pa          deg
 value    3.00E+02     80.        2.29E+05    -52.        6.20E-05    -52.
result     .00         .00         .00         .00         .00         .00
TARGET     10a         10b
 name  HARDE:R(1/Z HARDE:I(1/Z
 units
 value      .00         .00
result      .00         .00
Potential TARGETS still available:
Addr Seg:Par-Type    Current Value
 3a SOFTEN:Re(Z) =   .0000
 3b SOFTEN:Im(Z) =   .0000
 6e SXLASt:HeatIn=   .0000         W
```

One can think of these guesses and targets as paired up in the following way. The T-begin guess lets DeltaE hit the T-hot target; these two are so nearly equal, and so trivially related, that they could easily be dropped from the vectors if necessary. The two branch-impedance guesses and the two IMPEDance guesses let DeltaE reach four targets: the two p targets at the UNION and the two U free targets that essentially determine the displacer piston's motion. The heat removed at the hot heat exchanger determines the temperature at the cold heat exchanger. Finally, the two p guesses in the BEGIN segment allow DeltaE to achieve $U = 0$ at the HARDENd at the end of the apparatus.

Running this file produced the .out file listed above, and the .dat file below:

```
 -= Stirling cooler w displacer piston, illustrating TBRANCH--UNION          =-
 frequency=      55.000Hz     mean pressure=   2.000E+06Pa

  T(K)          p(Pa)               U(m^3/s)             hdot(W)   wdot(W)
   300.1    210877. -193152.0      0.00033  0.00005          29.38     29.38
!----------------------------------  1 ----------------------------------
```

```
 TBRANCH   branch to displacer
 TTTTTTTTTTTTTTTTTTTT Branching into Tee Level= 1 TTTTTTTTTTTTTTTTTTTT
   300.1    210877. -193152.0     -0.00004  0.00005          -8.74     -8.74
!----------------------------------  2 ----------------------------------
 IMPEDANCE displacer, sort of
 Imped. work (heat extracted)=   -1.64     Watts
   300.1    139875. -181568.9     -0.00004  0.00005          -7.11     -7.11
!----------------------------------  3 ----------------------------------
 SOFTEND   reconnect at UNION
 impedance (p A/rho a U)=(   -132.    ,   0.899    )

   300.1    139875. -181568.9     -0.00004  0.00005          -7.1     -7.11
 TTTTTTTTTTTTTTTTTTTT Returning to Trunk Level= 0 TTTTTTTTTTTTTTTTTTTT
   300.1    210877. -193152.0      0.00036  0.00000          38.12     38.12
!----------------------------------  4 ----------------------------------
 SXFRST    Hot heat exchanger
 Heat =     -36.051 (W)    metal temp=     300.000 Kelvin
   300.1    205749. -193181.4      0.00036  0.00000           2.07     37.19
!----------------------------------  5 ----------------------------------
 STKSC     regenerator
    80.0    140139. -181883.2      0.00004 -0.00005           2.07      7.13
!----------------------------------  6 ----------------------------------
 SXLASt    cold heat exch
 Heat =       5.033 (W)    metal temp=      80.000 Kelvin
    80.0    139875. -181568.9      0.00004 -0.00005           7.11      7.11
!----------------------------------  7 ----------------------------------
 UNION     displacer cold end
 Union P match difference=  -7.559E-03 Pa;  -1.890E-06 deg.
    80.0    139875. -181568.9      0.00000  0.00000           0.00      0.00
!----------------------------------  8 ----------------------------------
 FREETARGETdisplacer U
 FREET output =   6.200E-05
    80.0    139875. -181568.9      0.00000  0.00000           0.00      0.00
!----------------------------------  9 ----------------------------------
 FREETARGETdisplacer phase
 FREET output =   -52.0
    80.0    139875. -181568.9      0.00000  0.00000           0.00      0.00
!---------------------------------- 10 ----------------------------------
 HARDEND   the end
 inverse impedance (rho a U/p A)=(  -8.211E-09,   6.616E-09)

    80.0    139875. -181568.9      0.00000  0.00000           0.00      0.00
```

The user might next generate cooling power curves by using the cold temperature target as an independent plot variable and the cooling power as dependent plot variable; or the user might explore the frequency dependence of the cooler, by using frequency as an independent plot variable; or the user might want to add more realism to the model by including the large dead volumes shown in the figure near the pistons. If inertial and viscous effects are presumed negligible in those volumes, they can be modeled as `COMPLIANCEs`:

```
TITLE
BEGIN
COMPLIANCE
   TBRANCH
```

```
   IMPEDANCE (the displacer)
   SOFTEND
SXFRST
STKSCREEN
SXLAST
COMPLIANCE
UNION  ('connects' to softend above)
HARDEND
```

The user will soon discover that this is a surprisingly robust model, considering the large number of guesses and targets: the model tolerates steps in independent variables of several percent without getting lost.

`TBRANCH` and `UNION` are intended for duct networks, where temperature is constant and hence p_1 and U_1 are the variables of interest. For more complex systems, the segments `HBRANCH` and `HUNION` are energy-conserving versions of `TBRANCH` and `UNION`. Use them if you are branching at locations where $\dot{H} \neq \dot{W}$, such as at a branch to a second stage regenerator within a two-stage pulse tube refrigerator. `HBRANCH` incorporates a potential guess `Hfrac`, giving the fraction of the incoming enthalpy that goes into the branch. Use `Hfrac` as a guess to hit a target down the branch, such as a temperature. `HUNION` incorporates an additional potential target, that the temperature in the trunk at the union be equal to that at the associated branch end.

C Turbulence

A turbulence algorithm can be enabled in `ISODUCT`s, `INSDUCT`s, `ISOCONE`s, and `INSCONE`s by use of an otherwise hidden input parameter: parameter **d** (**f** for '`CONE`s), the relative roughness (roughness height divided by pipe diameter). Set the roughness equal to zero for smooth walls, or to some small value greater than zero for rough walls. To ensure a laminar calculation, set the roughness equal to -1 (which will cause the parameter to be hidden once again).

The turbulence algorithm follows the spirit of the assumptions of Iguchi *et. al.* [Bull JSME 25, 1398–1405 (1982)]. It assumes that oscillatory-flow losses can be calculated by using the Moody friction factor (valid for steady flow) at each instant of time during the oscillatory flow. This assumption has little experimental validation in the range of Reynolds number and R/δ_ν of interest in thermoacoustics, but we believe it provides a useful estimate, better than no estimate at all. For more details, see Chapter VI.

D Variable Gas Mixtures

Several binary mixtures of gases have proven useful in thermoacoustic devices because of their improved Prandtl numbers and the option to adjust the resonance by changing the sound speed. DELTAE's fluid library contains three such mixtures: He-Xe, He-Ar, and He-Ne. These fluids are specified by a string on a line after a segment's numerical parameters, as are other fluids, but the string contains a 5 character field that represents the fraction of helium in the mixture (for example, `0.981hexe` or `0.889hear`, containing 98.1% and 88.9% helium, respectively).

If all but the first of the fluids (in the `BEGIN` segment) are specified using `sameas 0` statements, it is possible to use the helium fraction of the mixture as an iteration variable for resonance tuning. Simply select `0h` from the `(u)se` menu option (it may instead be a plot variable, if you choose). In the `out` file, the fluid written out will reflect the final concentration used.

Our equations for He-Xe properties are not valid for Xe fractions in excess of 0.5.

E User-Defined Fluid/Solid

DELTAE has a provision that allows users to specify 'external' fluids or solids that are not part of its internal library of thermophysical properties. Properties are derived, according to current operating conditions, from Eqs. V.1, V.2 and V.3 (below) using coefficients read from a user-written text file. Up to five distinct external fluids and five external solids can be used at one time.

The file can have any name valid under the operating system under which DELTAE is running, and it should end with the extension `.tpf`. If the root filename is the same as any predefined fluids, DELTAE will replace its internal calculations for that fluid with those given in the user file. To request a user-defined fluid, simply use the root file name as you would any other fluid. The `.tpf` file should be in the same directory or folder as the model file. The name of the fluid is set to the root filename of the external fluid file.

The file format is similar to the segment definitions we have used in models described in previous chapters in that comment lines can be added with an initial '!' and blank lines are ignored. Each property is specified by a line containing 1–10 real coefficients which are read in as C_{0-9}, where unused trailing parameters are set to zero. It is critical that the properties be arranged as shown: ρ, c_p, K, a^2, and μ. We also need the ratio of specific

heats, γ, and the expansion coefficient β, but these are calculated internally from

$$\beta = -\frac{1}{\rho}\frac{\partial \rho}{\partial T} \quad \text{and} \quad \gamma - 1 = \frac{T\beta^2 a^2}{c_p}. \tag{V.1}$$

Each of the five properties is derived from its 10 coefficients using the following equation:

$$\text{property} = C_0 + C_1\frac{p_m}{T + p_m C_2} + C_3 T + C_4 T^2 + C_5 T^{C_6} + C_7 p_m^2 T^{C_8} + p_m C_9, \tag{V.2}$$

where T and p_m are the absolute temperature (K) and mean pressure (Pa) for each point at which a segment using the fluid is evaluated.

Equation V.2 is a compromise between simplicity and flexibility; it is intended for use in a variety of simple expressions for gases and liquids and has a uniform syntax for specifying all 5 properties. There is only a limited mean pressure dependence, suitable for nearly ideal gases; for more complicated mean pressure dependence, multiple `.tpf` files should be written for each mean pressure range used.

To illustrate the use of these coefficients, consider the example below. To replace the (ideal) helium gas in a model with a more accurate representation that calculates density and sound speed using the first coefficient of the virial expansion for helium, we can write the following file, call it `helium.tpf`, and put it in the same directory as our model:

```
! external fluid; He with first virial coeff for (B=12cc/mole)
! Equation is:
! C0 + C1*pm/(T+C2*pm) + C3*T + C4*T^2 + C5*T^C6 + pm^2 *C7*T^C8 + pm*C9
! Density, rho (m^3):
  0.  4.814e-4 1.44e-6
! isobaric heat capacity, cp (J/kg/K):
  5192.
! Thermal conductivity, k0 (W/m/K):
 0. 0. 0. 0. 0. 0.0025672  0.716
! Square of sound speed, a^2 (m^2/s^2):
  0.  0.  0.  3461.92  0. 0. 0. 0.  0. .0100
! Viscosity, mu (kg/s/m):
 0. 0. 0. 0. 0. 0.412e-6  0.68014
```

The coefficients for density were determined using

$$\rho = \frac{p_m M}{R(T + B p_m/R)},$$

where $R = 8.314$ J/mole-K, $M = .0040026$ kg/mole, and the first virial coefficient $B = 1.2 \times 10^{-5}\text{m}^3$/mole. We set $C_1 = M/R$ and $C_2 = B/R$. For squared sound speed, we need to satisfy

$$a^2 = \frac{\gamma RT}{M}\left(1 + 2\frac{Bp_m}{RT}\right),$$

so we set $C_3 = \gamma R/M$, and $C_9 = 2B\gamma/M$, where $\gamma = \frac{5}{3}$. See section C.1 in Chapter VI to compare this with how helium properties are calculated in DELTAE's internal routine.

For equations that cannot be manipulated to fit the format of Eq. V.2, we suggest generating a table of data near the expected operating conditions and using curve-fitting tools to generate appropriate coefficients.

User-defined solids follow an identical format, except that only the first three lines are required to specify ρ_s, c_s, and K_s. The meaning of coefficients C_1 and C_2 are also redefined to provide an exponential capability, so the equation for solids is

$$\text{property} = C_0 + C_1 \exp(-TC_2) + C_3T + C_4T^2 + C_5T^{C_6} + C_7p_m^2T^{C_8} + p_mC_9. \qquad \text{(V.3)}$$

It is a good idea to check each new external fluid or solid by using the `(t)hermophysical` command available in the main menu (external fluids or solids show up first and are selected with negative integers). Users can also insert the special `THERMophysical` segment using the fluid/solid to display the properties in the `.out` and `.dat` files, or to plot them (see below).

F Parameter Linking (Special Modes)

DELTAE is versatile in the way it uses different model parameters as guesses to meet its targets: length or volume (to achieve resonance at fixed frequency), stack length and position (to meet an efficiency and amplitude), or stack diameter (to get adequate power), for example. When such geometric variables are released to the solver for manipulation (or when they are made to change in a plot loop), there are often certain geometric relationships to other parameters that we would like to see maintained. For example, if the area of a duct increases, we must increase the associated perimeter as well. Another common wish is to lengthen a segment while simultaneously shortening another segment to keep overall length constant. Also, in a stack made of a constant thickness material in a duct of fixed diameter, we cannot blindly vary the pore size and expect the porosity to remain the same—this could lead to a misleading optimization if we are faced with these constraints. If we go to the trouble to calculate a porosity for our initial segment, we want DELTAE to respect it for the values it chooses as we run the model. 'Special modes' were introduced to link parameters for just these purposes.

A special modes dialog appears automatically whenever a parameter linking capability is possible for a variable that is chosen as a guess vector member:

```
MAIN: (rpwPncTCgudvomfst e?)> u
```

```
Guess/Target Address=? ( 0a) 4d
Selection: STKCIR:r0

Add to the guess vector (y/n)? y
Special modes can be  enabled as this parameter is varied
(Only one mode per segment possible):
Mode       Description
     0   Normal mode (no inter-related parameters)
    -1   Adjust porosity while y0(r0) varies (const. Area, L0)
Mode=? ( 0) -1
```

By selecting `-1` for the special mode, we have asked DeltaE to remember the following constant before it begins iterating:

$$\texttt{const} = \texttt{r0}^2/\texttt{poros} - \texttt{r0}^2$$

where `r0, poros` are the pore size and porosity of our initial stack. We assume that the effective plate material half-thickness is $\texttt{L0} = \texttt{r0}^2(1/\texttt{poros} - 1)$. During the iteration, as `r0` is changed, DeltaE assumes porosity changes as an ideal porosity would and calculates it from the following:

$$\texttt{poros} = \texttt{r0}^2/(\texttt{r0}^2 + \texttt{const}),$$

and the effective plate thickness is maintained.

If we create a plot varying the area of our first `INSDUct` (parameter 2a, in most of the examples of the previous chapters), the dialog looks like

```
MAIN: (rpwPncTCgudvomfst e?)> p
define plot variables.  One or two inputs (a-j)
and up to 10 outputs (A-J) can be plotted)
Plot Parameter Address=? ( 0A)2a

use for outer or inner (2d) plot loop (o/i)? o
Outer (or 1-D) Plot Loop:
Independent variable is  ISODUC:Area
Plot begins at ISODUC:Area  =  1.2920E-02    m^2
New value (<CR> to keep)=?
Plot ends at ISODUC:Area  =  1.2920E-02    m^2
New value (<CR> to keep)=? 2e-2
with a step of:    1.00
New value (<CR> to keep)=? 1e-3
Special modes can be  enabled as this parameter is varied
(Only one mode per segment possible):
Mode       Description
     0   Normal mode (no inter-related parameters)
    -2   Maintain consistent Perimeter as initial Area varies
Mode(n)=? ( 0)-2
```

By selecting `-2` for the special mode, we have asked DeltaE to remember the constant:

$$\texttt{const} = \texttt{perim}^2/\texttt{area}$$

and, later, to calculate the perimeter from

$$\texttt{perim} = \sqrt{\texttt{area} * \texttt{const}}.$$

This relationship keeps circular ducts circular and maintains the aspect ratio of rectangular ducts.

A very complicated example, even if somewhat confusing, can give some idea of the power of parameter linking. Interesting iterations can be done by using `sameas` parameters in combination with length parameter linking. For example, if segments 2 and 7 are `ISODUct`s, and segments 4 and 5 are `STKSLab`s of equal length (but different material or porosity, perhaps), we can iterate using stack length, keep these lengths equal, *and* keep the overall length and stack center position constant by doing the following:

1. For the length (`c`) of segment 5, specify `sameas 4c`.
2. `(u)se` parameter `4c` as a guess (you will have to clear another guess, or add a suitable target, to keep your guess and target vectors balanced).
3. When prompted to select a special mode for segment 4, choose '2' to keep the sum of segment 2 and 4's lengths constant.
4. Using the `(s)pecial modes editing` option, select parameter `5c` and set its mode to '7'.

If `4c` were an independent plot loop variable instead of a guess vector member, the procedure above would be identical, except that item (2) would be a `(p)lot` selection option instead of a `(u)se` dialog. The following is a list of all parameter linking modes and the segment types for which they are available:

`n` Keep Length + Length in segment (n) constant: All segments with length.

`0` Normal mode (no inter-related parameters): All segments

`-1` Adjust porosity while y0(r0) varies (const. Area, L0):] Most stacks and heat exchangers.

`-2` Maintain consistent Perimeter as initial Area varies: Ducts and cones.

`-3` Maintain consistent Perimeter as final Area varies: Cones.

`-4` Adjust porosity as L0 varies (const. Area, y0): `STKSLab`.

`-5` Maintain consistent surface area as volume varies: `COMPLiance`.

`-6` Maintain constant V & valid perim., area as length varies: `STKDUct`

-7 Vary imaginary part to preserve magnitude (where possible): IMPED, BRANCh, TBRANch, and HBRANch.

-8 Vary imaginary part to preserve phase angle:] same as -7.

G Thermophysical Properties

The (t)hermophysical menu selection (see Chapter VI for further details) allows the user to have keyboard access to the library of fluid and solid properties for a given state (which defaults to the current temperature, pressure, acoustic frequency, and fluid or solid). This feature has proven so convenient that we often start DELTAE simply to look up the transport properties of gases. (For this purpose, it is often useful to have a dummy file present (e.g,. nil.in) that contains only a TITLE line. If you respond to the input file prompt with this filename, DELTAE will quickly go to the menu line and allow you to access the options.)

A companion to the (t)hermophysical menu selection is THERMophysical segment type, which takes no input parameters except for the fluid and solid type (again, see Chapter VI for a summary). This segment can be inserted anywhere in a model where the user wants to know the fluid and solid properties at the local temperature and pressure, whatever they may be at the time. Both the .out and .dat files contain outputs for these properties where the segment is inserted. By using the plotting loops, tables of properties can be generated over ranges of temperature, pressure, or frequency by varying these values in a BEGIN segment, ending the model with a THERMophysical segment, and plotting as many of the outputs as are required.

H State Variable Plots

State variable plots allow you to view the distribution of temperature, pressure, and enthalpy along the entire length of a model. The format is somewhat similar to that of the *.dat file, but with more detail. Selecting (G)enerate state variable plot from the (E)xtras submenu will cause a *.spl file to be written. The output below was generated from the 5inch.in example file (section III.B) *before* guess and targets were added, and before iterations were performed:

```
 ->5INCH.spl
!Created@12:55:56 16-JUN-97 with DeltaE Vers. 3.6b3 for the Power Macintosh
 -= Five-Inch Thermoacoustic Engine                                    =-
Seg# x(m) GasA(m^2) T(K)    Re[p](Pa)  Im[p](Pa) Re[U]   Im[U](m^3/s) Hdot(W)
```

```
 1 0.000 0.012920  500.0    80000.0        0.0   0.00000   0.00000      0.00
 1 0.000 0.012920  500.0    80000.0        0.0  -0.00003   0.00000     -1.20
 2 0.000 0.012920  500.0    79992.8        0.1  -0.00006  -0.00790     -1.20
 2 0.056 0.012920  500.0    79935.6        0.6  -0.00011  -0.02371     -1.20
 2 0.112 0.012920  500.0    79821.2        1.6  -0.00016  -0.03949     -1.20
 2 0.167 0.012920  500.0    79649.7        3.0  -0.00021  -0.05525     -1.20
 2 0.223 0.012920  500.0    79421.2        4.7  -0.00026  -0.07096     -1.20
 2 0.279 0.012920  500.0    79285.7        5.8  -0.00029  -0.07880    -11.65
 3 0.279 0.005078  500.0    79285.7        5.8  -0.00029  -0.07880    -11.65
 3 0.339 0.005078  500.0    78318.1      403.1  -0.00184  -0.08697   2198.55
 4 0.339 0.010465  500.0    78318.1      403.1  -0.00184  -0.08697   2198.55
 4 0.395 0.010465  428.8    77731.7      847.5   0.00166  -0.09753   2198.55
 4 0.451 0.010465  375.9    76965.8     1289.3   0.00455  -0.10886   2198.55
 4 0.506 0.010465  334.6    76002.3     1733.3   0.00702  -0.12056   2198.55
 4 0.562 0.010465  300.9    74822.6     2182.1   0.00920  -0.13236   2198.55
 4 0.618 0.010465  272.8    73407.8     2636.4   0.01117  -0.14405   2198.55
 4 0.618 0.010465  272.8    73407.8     2636.4   0.01117  -0.14405   2198.55
 5 0.618 0.006158  272.8    73407.8     2636.4   0.01117  -0.14405   2198.55
 5 0.669 0.006158  272.8    71193.4     2996.3   0.01045  -0.15121    145.54
 6 0.669 0.012670  272.8    62577.4     2479.4   0.01377  -0.23637    145.54
 6 1.400 0.012670  272.8    35543.1     1109.6   0.01814  -0.36310    145.54
 6 2.130 0.012670  272.8      684.6     -470.2   0.01887  -0.40988    145.54
 6 2.861 0.012670  272.8   -34324.0    -1945.9   0.01584  -0.36643    145.54
 6 3.592 0.012670  272.8   -61775.6    -3026.3   0.00967  -0.24231    145.54
 6 4.323 0.012670  272.8   -70673.7    -3347.8   0.00579  -0.15799     60.00
 7 4.323 0.012670  272.8   -70673.7    -3347.8   0.00579  -0.15799     60.00
 7 4.323 0.012670  272.8   -70673.7    -3347.8   0.00580  -0.15799     59.46
```

The following features of state variable plots are noted:

- When generating a state variable plot, DeltaE does not iterate; it simply takes one pass through the model using the current guess variable values.
- During the pass, DeltaE prints `Nint`$/2 + 1$ (`Nint` is the number of Runge-Kutta steps—see section J for details) lines of data for each segment that it knows how to integrate (stacks, ducts, and cones).
- Two lines are printed for elements which do direct calculations (heat exchangers, endcaps, etc.): one before, and one after the segment is computed.
- Segments that do not have any physical effect, such as freetargets, `BEGIN`, and `'END` segments, generate no output in the plot.
- The third column, `GasA`, is the current cross-sectional area times the porosity of the segment.
- Work is not an output, but, in a spreadsheet, it can be derived from the plotted variables using $\dot{W} = \Re[p_1 \tilde{U}_1]/2$
- when a model contains a branch segment, a blank line will be left before and after the branch in the output. Also, the x distance counter begins at zero again in the branch.

I Geometry

When sizes are changing dynamically, it is often desirable to know something about the physical size and layout of a device under design, no matter how abstract the available information may be. DeltaE has an option to write a 'geometry' file for this purpose. Selecting it causes a `geo` file to be written that contains X-Y pairs suitable for plotting with your favorite graphics software. When this file is given to graphing software, the resulting plot is representative of a half cross-section of a cylindrical device similar to the model. The figure below is an annotated plot made from the geometry file for our final example of the Hofler refrigerator:

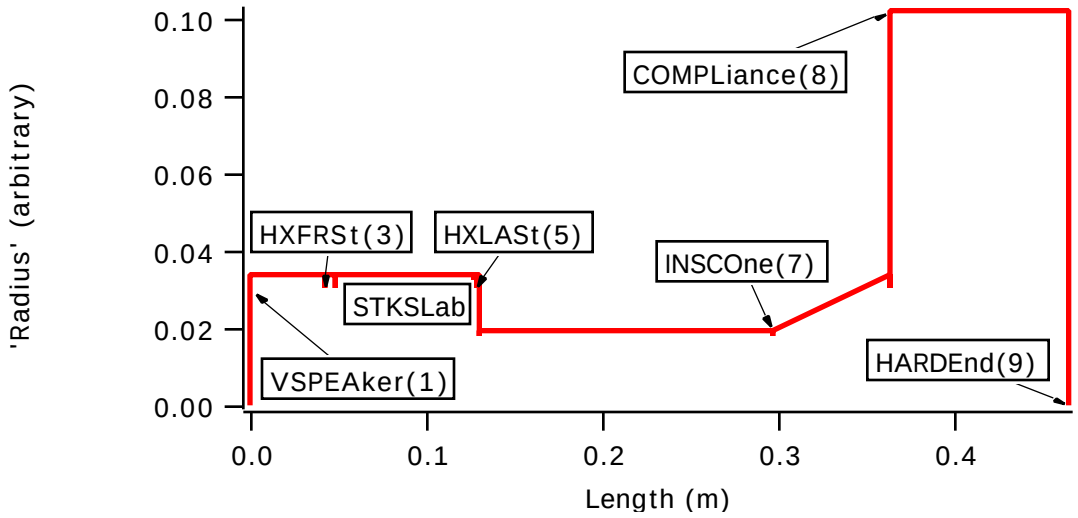


Figure V.3: Geometry of Hofler refrigerator example.

We have labeled all of the segments, except for the ducts, with their numbers and types. DeltaE generates little 'tick' marks to identify the breaks between segments. The lines down to zero on either end are generated by the `VSPEAker` and `HARDEnd` segments, respectively. The height at most points is proportional to the square root of the area. The `COMPLiance` is the exception; it looks nothing like Hofler's sphere. It is a symbolic cylinder that has length equal to radius (sort of—factors of π are ignored) proportional to the specified volume. (Some models, such as those with active branches, are not supported properly by geometry files yet.)

J Tuning and Debugging

The `(T)olerances/debugging` menu selection gives the user access to a number of internal parameters that control the quantity of output and diagnostic information generated and the way that the solver approaches the iterations it will perform. An explanation of these parameters is given below:

`Nprint` If `Nprint` ≤ 0, the `.dat` file will contain only the final converged iteration of the model. Otherwise, DeltaE saves every `Nprint`th intermediate iteration. If `Nprint` ≥ 0, intermediate steps in every stack integration are included in the data file. For `Nprint` > 0, every segment is displayed to the screen (equivalent to typing the `.out` file). This can be useful in finding model errors that cause DeltaE to crash before the first converged data point is ever stored. If `Nprint` < 0, a concise iteration summary line is printed every `-Nprint+1` intermediate iterations. By setting `Nprint` to a larger negative integer, time-consuming screen output can be reduced, which will make calculations run several times faster on machines with good floating point performance. The summary line contains only the iteration number and the root-mean-square sum of the errors (targets − results), and the line will overstrike itself. If `Nprint` $= 0$, the iteration count and the complete guess and (target−result) vectors are displayed on sequential lines. Default: -1.

`PlotDat` This variable controls output generated during plots, where multiple solutions are processed sequentially. If `PlotDat` ≥ 0, all error messages are that occur when DeltaE has doubts about the convergence of a datapoint are announced (on a Mac-Intosh, 'OK' must be clicked in the alert box before calculations will continue. For other values of `PlotDat`, DeltaE will continue silently, but will still write the messages to the `.dat` file and mark the lines in the `.plt` file with a '$*$.' If `PlotDat` ≥ 1, all converged endpoints are written to the .dat file (it can become quite large!). For `PlotDat` $= 0$, only the most recent is kept. Default: 0.

`tolerance` Recommended range: 1.5e-7–1.e-2. This value governs the point at which DeltaE considers its iterations finished. The default value is close to the limit that can be reached using single-precision arithmetic (all DeltaE calculations are double precision). This value does not relate directly to errors between any particular result and its target value; it concerns changes in the norm of the error vector. Default: 1.00E-05.

`Runge-Kutta steps` This is an even integer that determines the number of integration steps used to span each stack-type segment, turbulent duct, or cone. It does not affect other segment types. It also determines the resolution with which state variable plots (the `(G)enerate` option described in the preceding section) are printed: `Nint`/2 lines per segment. Larger values will cause a slower, more accurate computation.

Small values will increase speed at the price of integration accuracy, but may cause convergence problems if the specified tolerance is too small. Output from every other step can be enabled with the `Nprint` parameter. Default: 10.

`Normalization mode` In a numerical problem where all of the input variables in the guess vector and all of the output variables used in the target vector are of wildly different magnitudes, a difficulty arises in choosing how much to change each variable and how much to weigh the errors between the target and the result values. Particularly, this affects `HARD-` and `SOFTEnd` segments. A 0.01 K error in a heat exchanger temperature is fairly benign to us, but in the complex end impedance, an error of 0.01 could leave us with hundreds of watts of power flow where there must be zero. In the standard mode (1), DeltaE uses the solver's internal method to normalize the solution vector, which usually does a reasonable job. For pathological cases, DeltaE has a special mode (2) that tries to normalize all input variables and output variable differences to unity. This can present its own problems, however, since we do not know how to normalize zero input variables (phases are a special case, automatically normalized by 360^o). In normalizing outputs, problems can occur if the model is very far from being converged, giving large initial error values; if it is very close to being converged, the errors could be near zero, presenting the other problem. Use mode 1 whenever possible, and mode 2 when you must. It may sometimes help to specify a zero input (target or guess) variable as some tolerably small nonzero number when using this mode. Default: 1.

`Step bound factor` recommended range: 0.01–100. This value regulates the size of initial excursions DeltaE makes from initial guesses to find the directions in which it must iterate. Some difficult cases can benefit by reducing this value. Default: 100.0.

`FCNerr` There is a limit to the accuracy with which a computer can calculate the 'function' that represents one complete pass through a model. The assumed value of this error affects the increments between iterations that the solver will choose; if the increments are too small, the effect on the result will be unpredictable. Larger values of `FCNerr` can speed iterations, with a less accurate endpoint. Too small a value can cause the solver to lose its way completely. This quantity is system-dependent, and it may be necessary to increase it slightly for very complex models. Recommended range: > 5.e-15. Default: 1.00E-10.

`Minimum Temperature` There is a temperature floor, 10 K by default, to prevent DeltaE's solver from exploring unphysical temperatures such as negative temperatures. Brave users with special needs at lower temperatures (and generally with their own, external thermophysical properties files!) can set this floor to a lower value. (Some of DeltaE's internal fluids use a higher temperature floor. He-Ar mix, for example, does not calculate properties below 70 K in order to prevent unreasonable values from being generated).

`Plot field delimiter` Normally, DeltaE generates plots and state variable plot files with the numbers in fixed with columns. Some spreadsheets and plotting packages do not process these files as well as they do delimited text. If `plot_f_sep` is set to —texttt1, a comma will be placed between columns in each of these plot types.

J.1 Initialization files.

If any of the above parameters are modified from their default values, you will generally want to keep the new values for every new run on the current model, and reuse them every time you execute the program. Therefore, whenever the `(T)olerances/debugging` option is used to change the default settings, all of the tunable parameters are written to a special file when the model is saved. This file has the same base filename as the model, with the extension `ini`. Whenever a new model is loaded, DeltaE checks for a `.ini` file in the same directory with a matching base filename and loads these settings if it is found. This file is written in `NAMELIST` format which makes it easy to examine and modify using any text editor.

Frequently, a collection of similar models will reside in a single subdirectory, and these files will share identical custom settings. For these situations, any `.ini` file can be copied (or renamed) to `default.ini` and DeltaE will use the settings within it for any model run from the same directory, If a model has its own individual `.ini`, its settings will take precedence.

Chapter VI

Reference

A General

DeltaE solves the one-dimensional wave equation, with temperature evolution, in the usual low-amplitude, "acoustic" approximation. It does not allow superposed steady flow, nor does it include nonlinear effects.

In each pass, DeltaE integrates from `BEGIN`ning to `HARDEnd` or `SOFTEnd`, with respect to 5 real variables: real $T_m(x)$, complex $p_1(x)$, and complex $U_1(x)$. It uses the differential (or simpler) equations appropriate for each segment, with the evolution of these variables in each segment controlled by local parameters, such as geometry and energy flow, and global parameters, such as frequency and mean pressure. Continuity of T_m, p_1, and U_1 are used at the junctions between segments.

In general, a pass of DeltaE's integration does not result in desired values of all variables. A shooting method is used to adjust chosen initial variables, called 'guesses," in order to hit desired results, called "targets." Initial guesses are provided by the user, or (more commonly) by a previous run of DeltaE.

The table below serves as a guide to choice of guess and target variables:

	Variables *we* think of as fixed (including independent variables in plots)	Variables *we* think of as results (any of these can be plot results)
Inputs for each pass of DELTAE. Includes T-begin, p-begin, U-begin, p-mean, freq, all dimensions, 'ducer coefficients, volts @`VDUCEr`, heat @`HXFRSt, HXMIDl`, gas concentration.	simply fixed in input file	guess
Results from each pass of DELTAE. Includes all T, p, U except in `BEGIN`; heat @`HXLASt`; current in `VDUCEr`s; combinations of above such as `COPRTarg`, `DIFFTarg`, Z at ends.	target	simply results in `.out` and `.dat` files

B Segments

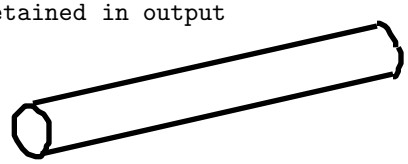
All of DELTAE's segment types are listed by functional grouping in this section. An alphabetical listing and cross-reference is presented at the end of the section.

B.1 Ducts, cones

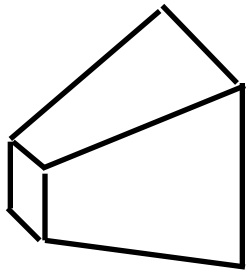
Segment types: `ISODUct, INSDUct, ISOCOne, INSCOne`

Sample input-file segments:

```
ISODUCT    comments typed here are retained in output
3.14e-4  m2 area
0.0628   m perim
0.1      m length
helium     gas
copper     solid
```



```
ISOCONE2    Finite cone of square
ideal       solid
```



`INSDUct` and `INSCOne` use same formats as `ISODUct` and `ISOCOne`, respectively.

Use:

Use for ducts and cones of any cross-sectional shape (*e.g.*, square, circular) by giving suitable area and perimeter.

Computation algorithm:

In ducts, $p_1(x)$ propagates according to

$$\begin{aligned} p_{out}(x) &= p_{in}\cos kx + (p'_{in}/k)\sin kx, \\ p'_{out}(x) &= -kp_{in}\sin kx + p'_{in}\cos kx, \text{ where } p' = dp/dx. \end{aligned} \tag{VI.1}$$

with complex wavevector k, given by

$$k = \frac{\omega}{a}\sqrt{\frac{1+(\gamma-1)f_\kappa/(1+\epsilon_s)}{1-f_\nu}}. \tag{VI.2}$$

A cylindrical geometry is assumed for f_κ and f_ν so that

$$f_\kappa = \frac{2J_1[(i-1)r_0/\delta_\kappa]}{(i-1)(r_0/\delta_\kappa)J_0[(i-1)r_0/\delta_\kappa]}, \quad f_\nu = \frac{2J_1[(i-1)r_0/\delta_\kappa]}{(i-1)(r_0/\delta_\kappa)J_0[(i-1)r_0/\delta_\kappa]}. \tag{VI.3}$$

Also,

$$\epsilon_s = \left(\frac{K\rho_m c_p}{K_s\rho_s c_s}\right)^{1/2}. \tag{VI.4}$$

In cones, p_1 propagates according to the lossy Webster horn equation:

$$\left[1+\frac{\gamma-1}{1+\epsilon_s}f_\kappa\right]p_1 + \frac{a^2}{\omega^2}\frac{1}{A}\frac{d}{dx}\left[(1-f_\nu)\,A\frac{dp_1}{dx}\right] = 0. \tag{VI.5}$$

The perimeter varies linearly from its initial value to its final value; area varies quadratically. Hence, circular cones have circular cross-sections everywhere, with diameter varying linearly with axial position.

In very narrow cones (or ducts), f_κ and f_ν are calculated using complex Bessel functions for $R/\delta < 25 (Eq.\ VI.3)$. Where $R/\delta > 30$, the boundary-layer approximation is used. For intermediate values, linear interpolation is used to make a smooth match between the two regimes. While the narrow duct solution assumes a circular cross-section, the shape of the duct is irrelevant in the boundary-layer approximation. A square duct with dimensions much larger than pentetration depth can be modeled simply by choosing perimeter = $4\sqrt{\text{area}}$, for example.

In `ISO`thermal ducts and cones, $\dot{H} = \dot{W}$ everywhere. This assumes that the duct/cone is thermally anchored, so power dissipation is carried away externally. Thermoacoustic heat transport along the perimeter, which in fact contributes a small difference between $\dot{H}$ and $\dot{W}$, is neglected.

In `INS`ulated ducts and cones, the power dissipation is deposited in the adjacent heat exchanger. If several `INSDUct`s and/or `INSCOne`s are strung together, the power dissipated in all of them should show up in the nearest heat exchanger. This feature of DeltaE is not yet fully bugproof. For example, a `BRANCh` between a heat exchanger and `INSDUct` ruins the thermal link; initial `ENDCAp`s do not share the thermal link to the heat exchangers; and nonzero real targets in `SOFTEnd` and `HARDEnd` cause `INSDUct` to give nonsense. Use with caution. If results look unreasonable, they are.

(See also `STKDUct`, which allows a temperature gradient along a duct. It is described under Stacks below.)

Turbulence extensions:

DeltaE's turbulence algorithm assumes that turbulent oscillatory flow is described by the Moody friction factor at each instant of time during the oscillatory flow. [See any engineering fluid mechanics textbook to review the Moody friction factor as a function of Reynolds number and relative roughness of the pipe wall.] This assumption must be excellent in the low frequency limit, in which $R/\delta_\nu \longrightarrow 0$. This limit is approached in many inertances for pulse tube refrigerators. We do not know how good the assumption is for large R/δ_ν, which is of interest in the resonators of standing-wave thermoacoustic systems. For experimental validation of the assumption for intermediate R/δ_ν, see Iguchi et al., Bull. JSME **25**, 1398–1405 (1982).

DeltaE's turbulence algorithm can be enabled only in `ISODUCT`s, `INSDUCT`s, `ISOCONE`s, and `INSCONE`s. To do so, include (or `(m)odify`, from within the program) parameter **d** in the '`DUCT` segment in the input/output file (use parameter **f** for '`CONE`s). Parameter **d** is the relative roughness ε, whose definition can be found in fluid mech textbooks: roughness height divided by pipe diameter. A typical value might be 10^{-3}. Setting this parameter

equal to minus one makes that line of the input/output file disappear, returning the `'CONE` calculation to laminar.

A sample of a modified duct segment, with turbulence enabled, is given below. (For comparison, this is the final duct of the "5-inch engine" that was given in Section III.B. The area of the duct has been reduced so that it becomes turbulent over most of its length.)

```
!-------------------------------- 6 --------------------------------
 ISODUCT    Cold Duct
  4.0000E-03 a Area      m^2             1.1129E+05 A |p|       Pa
  0.2220     b Perim      m    (-2)       -176.6     B Ph(p)     deg
   3.650     c Length     m               2.7085E-05 C |U|     m^3/s
  1.0000E-04 d Srough                    -176.7     D Ph(U)     deg
                                          1.507     E Hdot      W
 helium     Gas type                     1.507     F Work      W
```

The portion of the `.dat` file corresponding to the above duct segment is as follows:

```
!----------------------------------  6 ----------------------------------
 ISODUCT   Cold Duct
 Re=0.29E+06, r/dn= 210.7, m=  1.1574, m-prime=0.9987 at start;
 Re=0.39E+06, r/dn= 210.7, m=  1.4568, m-prime=0.9970 peak @x= 1.2775
  End of this segment is laminar.
 Heat extracted:    137.     Watts
   306.6   -111099.   -6592.1     -0.00003  0.00000           1.51      1.51
```

We note that losses in this duct have increased to 137 W ("Heat extracted"), but there are also three new lines in the listing. The parameters given in the first two lines are Reynolds number (referenced to diameter), the ratio of radius to viscous penetration depth δ_ν, the dissipation multiplier m, the inertial multiplier m' (see below), and the location along the duct. The third line says that velocities return to a laminar regime before the end of the duct. If the peak Reynolds number occurs at either end of the duct, or if the entire duct is laminar, the middle line detailing the peak location will be omitted.

The volumetric velocity and hence Reynolds number N_R vary sinusoidally in time; hence, the instantaneous Moody friction factor f_M has a complicated time dependence. We simplify this time dependence by essentially using a Taylor-series expansion around the peak Reynolds number:

$$f_M(t) \simeq f_M + \frac{df_M}{dN_R}\frac{N_R}{|U_1|}\left(\mathrm{Re}\left[\mathrm{U_1 e^{i\omega t}}\right] - |\mathrm{U_1}|\right), \tag{VI.6}$$

where f_M and the derivative on the right-hand side are evaluated at the peak Reynolds number. It is then straightforward to integrate the instantaneous power dissipation over a

full cycle, obtaining for the time-averaged power dissipation per unit length

$$\frac{d\overline{\dot{E}}}{dx} = \frac{\rho\left|U_1\right|^3}{3\pi^3 R^5}\left[f_M - (1-9\pi/32)\,N_R\frac{df_M}{dN_R}\right], \tag{VI.7}$$

where the quantities in the square bracket are evaluated at the peak Reynolds number.

When this is compared to the equivalent result for laminar flow

$$\frac{d\overline{\dot{E}}}{dx} = \frac{\rho\left|U_1\right|^2\omega}{2\pi R^2}\mathrm{Re}\left[\frac{\mathrm{i}}{1-\mathrm{f}_\nu}\right], \tag{VI.8}$$

it is apparent that turbulence multiplies the dissipation by a factor m given by the ratio of the two expressions above:

$$m = \frac{\delta_\nu^2 N_R}{6\pi R^2}\frac{\left[f_M - (1-9\pi/32)\,N_R\,df_M/dN_R\right]}{\mathrm{Re}\left[\mathrm{i}/\left(1-\mathrm{f}_\nu\right)\right]}. \tag{VI.9}$$

DeltaE evaluates f_M and df_M/dN_R as a function of Reynolds number and ε using the iterative expression

$$\frac{1}{\sqrt{f_M}} = 1.74 - 2\,\log_{10}\left(2\varepsilon + \frac{18.7}{N_R\sqrt{f_M}}\right), \tag{VI.10}$$

which is a remarkably good approximation to the Moody friction factor [R. M. Olson, Essentials of Engineering Fluid Mechanics]. To account for turbulence, DeltaE increases the resistive component of the pressure gradient, and hence the viscous power dissipation, by m. It decreases the inertial pressure gradient by

$$m' = \left(\frac{1-\delta_\nu/R}{1-\delta_\nu/mR}\right)^2 \tag{VI.11}$$

to correct approximately for the steeper velocity gradient at the wall, which increases the effective area open to gas contributing to inertial effects. It also multiplies the thermal penetration depth by m, in an attempt to account very approximately for changes in thermal relaxation losses due to increased heat transfer. Both m and m' are displayed in the `*.dat` file.

At low enough velocities, $m \longrightarrow 1$ and DeltaE reverts to a laminar calculation. The $m = 1$ boundary between laminar and turbulent zones in DeltaE occurs roughly at

$$N_R \simeq 2000 \text{ for } R/\delta_\nu < 2, \tag{VI.12}$$

$$\frac{N_R}{R/\delta_\nu} \simeq 1000 \text{ for } R/\delta_\nu > 2. \tag{VI.13}$$

DELTAE versions prior to 3.3 had a simpler turbulence algorithm, which was adequate for standing-wave resonators but not for the Reynolds numbers and R/δ_ν's found in inertances for pulse-tube refrigerators. That algorithm was enabled by setting perimeters negative instead of positive in 'CONES. If you still have old DELTAE output files with negative perimeters, the current version of DELTAE should be able to read and interpret them; it will save them in the new format.

B.2 Lumped elements: compliance, endcap, impedance

Segment types: COMPLiance, ENDCAp, IMPEDance

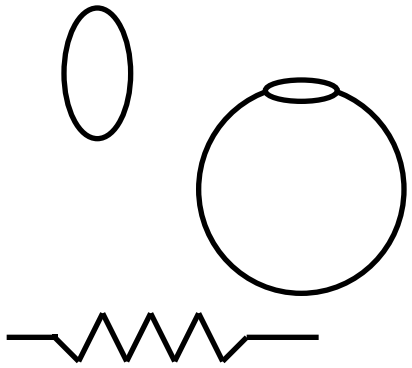
Sample input-file segments:

```
ENDCAP    a surface with thermal dissipation
1.134e-3 m2  Area
SAMEAS 0     Gas
ideal        solid

COMPLIANCE  this one is a sphere
0.1257  m2  Area
4.19e-3 m3  Volume
0.859hexe   Gas
nickel      solid

IMPEDANCE    just a lumped series impedance
1.0 Pa-s/m3 Re(Z)
-0.2 Pa-s/m3 Im(Z)
helium

! Blank lines at "solid" location are interpreted as "ideal" solid
```



Use:

An endcap is a surface area with thermal dissipation. It always absorbs work. A compliance is exactly that: a lumped acoustic volume element with surface thermal dissipation. An impedance is a lumped series complex impedance.

Computation algorithms:

An endcap does not affect pressure amplitude; volumetric flow changes according to

$$U_{out} = U_{in} - \frac{\omega p}{\rho a^2}\frac{\gamma - 1}{1 + \epsilon_s} A \frac{\delta_\kappa}{2}, \text{where} \quad \epsilon_s = \left(\frac{K \rho_m c_p}{K_s \rho_s c_s}\right)^{1/2}. \tag{VI.14}$$

Pressure p_1 is unchanged by a compliance; volumetric flow changes according to

$$U_{out} = U_{in} - i\frac{\omega p}{\rho a^2}\left[V - i\frac{\gamma - 1}{1 + \epsilon_s}A\frac{\delta_\kappa}{2}\right]. \tag{VI.15}$$

At an impedance, volumetric velocity is unchanged; pressure changes according to $p_{out} = p_{in} - ZU$.

B.3 Transducers, branches

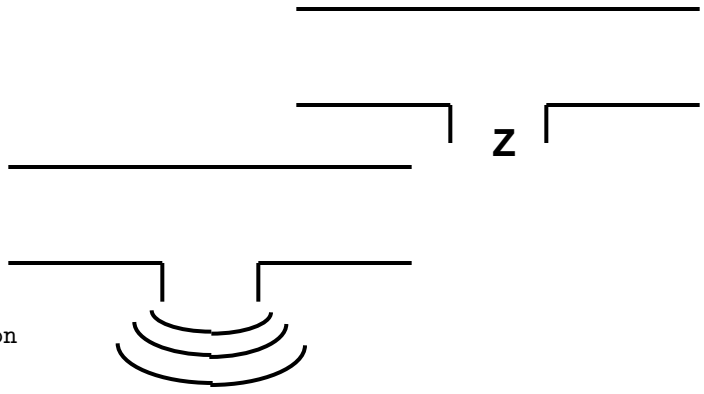
Segment types: `BRANCh, OPNBRanch, PISTBranch, VDUCEr, IDUCEr, VSPEAker, ISPEAker, VEDUCer, IEDUCer, VESPEaker, IESPEaker`

Sample input-file segments:

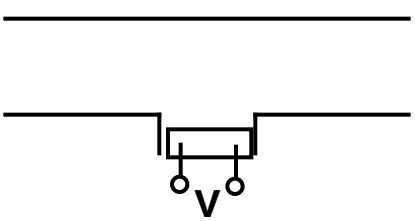
```
BRANCH
1  Pa-s/m3  Re(Z)
1.  Pa-s/m3  Im(Z)
0.500hear
ideal

OPNBRANCH
.05  Pa-s/m  Re(Z)/k^2
.2   Pa-s/m2 Im(Z)/k
air

PISTBRAN     Baffled Piston
.05  Radius   m
air
```

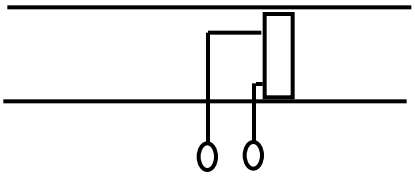



```
 VDUCER
   1.000E-09 a Re(Ze)   ohms
    .000     b Im(Ze)   ohms
   1.000E+04 c Re(T1) V-s/m^3
    .000     d Im(T2) V-s/m^3
  -1.000E+04 e Re(T2)   Pa/A
    .000     f Im(T2)   Pa/A
   1.000E-09 g Re(Zm) Pa-s/m^3
   1.000E-09 h Im(Zm) Pa-s/m^3
   10.0      i AplVol    V
 SAMEAS  0  Gas type
 ideal      Solid type
```




```
 VEDUCER     Enclosed driver
   1.000E-09 a Re(Ze)   ohms
    .000     b Im(Ze)   ohms
   1.000E+04 c Re(T1) V-s/m^3
    .000     d Im(T2) V-s/m^3
```

```
 -1.000E+04 e Re(T2)   Pa/A
   .000     f Im(T2)   Pa/A
  1.000E-09 g Re(Zm) Pa-s/m^3
  1.000E-09 h Im(Zm) Pa-s/m^3
  10.0      i  Vin      V
  45.0      j Ph(Vin)  deg
SAMEAS  0  Gas type
ideal      Solid type
```



IDUCEr or IEDUCer: same as VDUCEr or VEDUCer, except that current appears in line i (and phase of it on line j for enclosed units) instead of voltage.

```
VSPEAKER
  6.000E-04 a Area      m^2
   6.00     b   R       ohms
   .000     c L          H
   8.00     d B x L     T-m
  5.000E-03 e   M       kg
   .000     f   K       N/m
   .000     g   Rm     N-s/m
  -22.5     h AplVol     V
SAMEAS  0  Gas type
ideal      Solid type


VESPEAKER
  6.000E-04 a Area      m^2
   6.00     b   R       ohms
   .000     c L          H
   8.00     d B x L     T-m
  5.000E-03 e   M       kg
   .000     f   K       N/m
   .000     g   Rm     N-s/m
  62.0      h  Vin       V
 -37.2      i Ph(Vin)  deg
SAMEAS  0  Gas type
ideal      Solid type
```

ISPEAker or IESPEaker: same as VSPEAker or VESPEaker, except that current appears in line h (and phase of it on line i for enclosed units) instead of voltage.

Use:

BRANCh, OPNBRanch, and PISTBranch are side branches with fixed impedances. With BRANCh, the user specifies the real and imaginary parts of the impedance, assumed independent of frequency. OPNBRanch and PISTBranch incorporate the frequency dependence of radiation impedance. Thus radiation impedance at the end of an open tube radiating to 4π solid angle can be modeled as an OPNBRanch followed immediately by a HARDEnd.

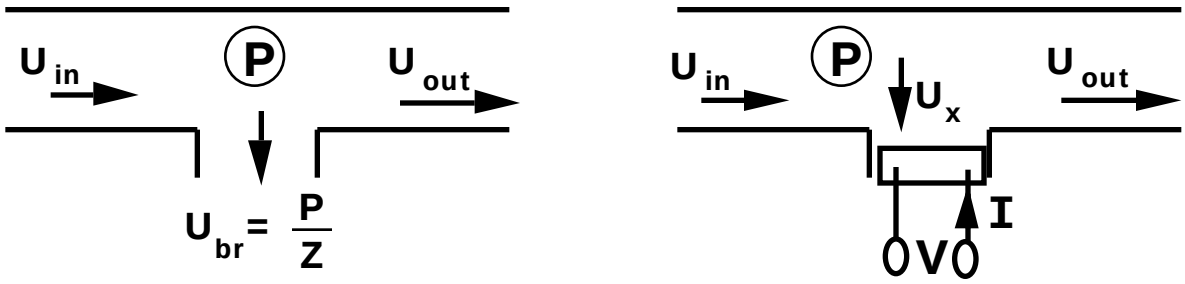


Figure VI.1: `BRANCH` (left) and branched `'DUCER` or `'SPEAKER` (right).

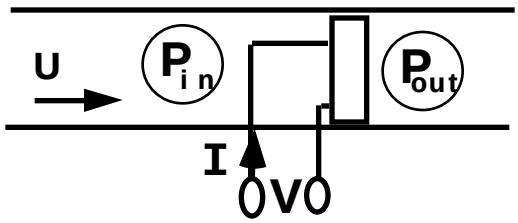


Figure VI.2: Enclosed `'EDUCer` or `'ESPEaker`.

`PISTBran` approximates the radiation impedance of a baffled piston of the given radius in radiating into the specified fluid.

The `'DUCEr`s and `'SPEAker`s are electroacoustic transducers. `'DUCEr`s have frequency-independent parameters; `'SPEAker`s let the user specify mass, spring constant, force constant, resistance, and inductance, so that frequency-dependent (even resonant) transducers can be modeled. With `IDUCEr` and `ISPEAker`, the user specifies the (real) current, and each pass of DeltaE calculates the (complex) voltage; with `VDUCEr` and `VSPEAker`, the user specifies voltage, and DeltaE computes current. `IDUCEr` and `ISPEAker` cannot be used with zero mechanical impedance because this would lead to a division of zero by zero (see below). Hence, use `VDUCEr` or `VSPEAker` for resonant or massless-and-springless transducers.

`'SPEAker`-type segments incorporate dissipation losses over their area as if they included an `ENDCAp`, but `'DUCEr`-type segments, which have no area parameter, do not. Enclosed `'ESPEaker`-type segments include dissipation losses for both sides of the driver.

Branched transducer elements, which require only magnitude of voltage or current applied as an input, effectively anchor the phase to zero for that parameter. The phase of pressure or velocity, as given in the `BEGIN` statement, must be varied in accordance with this reference. This effectively limits a model to only one `V` or `ISPEAker` (or `'DUCEr`), unless they are exactly in phase. Enclosed transducers, however, have a phase input which allows

them to be used together, or where the phase reference is determined by a `BEGIN` statement, for example.

Computation algorithms:

A branch is a side branch with complex impedance Z. Pressure is unchanged, but volumetric velocity changes according to $U_{out} = U_{in} - p/Z$. For an open branch, the numbers in the input file are multiplied by $(\omega/a)^2$ and ω/a respectively to obtain the impedance. For a baffled piston of radius r where the wavenumber is $k = \omega/a$ locally, the `PISTBran` radiation impedance is given by

$$Z_{\rm rad} = \frac{\rho a}{A}\left((1 - \frac{2J_1(2kr)}{2kr} + i\begin{cases} \left(\frac{4/\pi}{2kr} + \frac{\sqrt{8/\pi}\sin(2kr - 3\pi/4)}{(2kr)^{3/2}}\right) & \text{If } 2kr > 2.68 \\ \frac{(4/\pi)2kr}{3}\left(1 - \frac{(2kr)^2}{15}\right) & \text{otherwise} \end{cases}\right) \qquad \text{(VI.16)}$$

A branched transducer `IDUCEr, VDUCEr, ISPEAKer, VSPEAKer` is an object attached as shown in the figure like a branch impedance, but obeying the complex equations $V = Z_e I + T_1 U_x$, $p = T_2 I + Z_m U_x$. Pressure is unchanged, but volumetric velocity changes according to $U_{\rm out} = U_{\rm in} - U_x$.

There are three cases of interest for a branched transducer:

1. If an electrical load impedance $Z_{\rm ext}$ is hung on the transducer, it should be covered using a `BRANCH` segment, with $Z_{\rm branch} = p/U = Z_m - T_1 T_2/(Z_e + Z_{\rm ext})$.
2. If current I is given (and we take its phase to be real), then $U_x = (p - T_2 I)/Z_m$ and $V = Z_e I + T_1 U_x$.
3. If voltage V is given (and we take its phase to be real), then $I = (Z_m V - T_1 p)/(Z_e Z_m - T_1 T_2)$ and $U_x = (V - Z_e I)/T_1$.

An enclosed transducer `IEDUCer, VEDUCer, IESPEAker, VESPEAker` is an object attached in series with other segments, as shown in Fig. VI.2. Volumetric velocity is nearly unchanged (except for surface thermal losses—see below), but pressure is changed by the force exerted by the transducer, obeying the complex equations $V = Z_e I - T_1 U_1$, $p_{\rm out} - p_{\rm in} = T_2 I - Z_m U_1$.

There are three cases of interest for an enclosed (series) transducer:

1. If an electrical load impedance $Z_{\rm ext}$ is hung on the transducer, it should be covered using an `IMPEDANCE` segment, with $Z_{\rm imp} = Z_m - T_1 T_2/(Z_e + Z_{\rm ext})$.

2. If current I is given, then $p_{\text{out}} = p_{\text{in}} + T_2 I - Z_m U_1$ and $V = Z_e I - T_1 U_1$.

3. If voltage V is given, then $I = (V + T_1 U_1)/Z_e$ and $p_{\text{out}} = p_{\text{in}} + T_2 I - Z_m U_1$.

In the case of speakers, $Z_e = R + j\omega L$; $T_1 = -T_2 = Bl/A$; $Z_m = R_m/A^2 + j(\omega m - k/\omega)/A^2$. Thermal surface losses are computed for area A using the same formula as for an `ENDCAp`. Branch speakers are assumed to have area A exposed to the oscillating pressure. Enclosed speakers have area A exposed to p_{in} and area A exposed to p_{out}, because typically both sides of the speaker experience oscillatory pressure. As described above for `ENDCAps`, thermal surface losses manifest themselves as a small change in volumetric velocity.

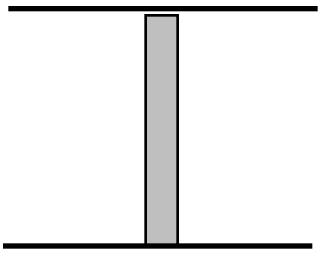
Note that `IDUCEr` and `ISPEAker` will crash if Z_m is zero, so it is best to use `VDUCEr` or `VSPEAker` for mechanically ideal or resonant transducers.

B.4 Heat exchangers

Segment types: `HXFRSt, HXMIDl, HXLASt; TXFRSt, TXMIDl, TXLASt; SXFRSt, SXMIDl, SXLASt`

Sample input-file segments:

```
HXFRST    parallel-plate heat exchanger
sameas 1  Area
0.600     GasA/A
6.35e-3 m Length
1.9e-4  m y0 = half of plate spacing
-20.0   W HeatIn
300.    K Est-T
sameas 0  Gas
copper    solid
```



```
 TXFRST     tube-in-shell heat exchanger
     0.2     a Area     m^2
    .188     b GasA/A
    .400     c Length    m
  6.350E-03 d radius    m   (radius of each tube)
  1.818E+05 e HeatIn    W
  1.000E+03 f Est-T     K
 sameas  0  Gas type
 nickel     Solid type


 SXFRST     Hot heat exchanger
  1.029E-03 a Area      m^2  total cross sectional area
  0.690     b VolPor         volumetric porosity
```

```
  2.000E-02 c Length     m
  6.450E-05 d   r_H      m   hydraulic radius
  -284.     e HeatIn     W
   300.     f Est-T      K     (t)
helium     Gas type
copper     Solid type
```

`*XMIDl` and `*XLASt` use same format.

Use:

Heat exchangers are used to inject or remove heat. They necessarily have surface area, so they experience both viscous and thermal dissipation of acoustic power. In `HX...`s the thermoacoustic working fluid is between parallel plates; in `TX...`s it is inside cylindrical tubes; and in `SX...`s the geometry is randomly-stacked screens.

Heat exchangers have a temperature difference between metal temperature and fluid mean temperature that is proportional to the heat flow. In `HX...`s and `TX...`s the proportionality constant is not well verified experimentally; we believe it to within a factor of 2.

`SX...`s are valid only for hydraulic radius smaller than thermal and viscous penetration depths. There is no warning if this bound is exceeded.

Computation algorithms:

In `HX...` and `TX...` heat exchangers, $p_1(x)$ propagates according to

$$\begin{aligned} p_{out}(x) &= p_{in}\cos kx + (p'_{in}/k)\sin kx, \\ p'_{out}(x) &= -kp_{in}\sin kx + p'_{in}\cos kx, \text{ where } p' = dp/dx, \end{aligned} \tag{VI.17}$$

with complex wavevector k, given by

$$k = \frac{\omega}{a}\sqrt{\frac{1+(\gamma-1)f_\kappa/(1+\epsilon_s)}{1-f_\nu}}. \tag{VI.18}$$

`HX...`s use parallel plate geometry in computing f_κ, f_ν, and ϵ_s:

$$\begin{aligned} f_\kappa &= \frac{\tanh[(1+i)y_0/\delta_\kappa]}{(1+i)y_0/\delta_\kappa}, \quad f_\nu = \frac{\tanh[(1+i)y_0/\delta_\nu]}{(1+i)y_0/\delta_\nu}, \\ \epsilon_s &= \left(\frac{K\rho_m c_p}{K_s\rho_s c_s}\right)^{1/2}\frac{\tanh[(1+i)y_0/\delta_\kappa]}{\tanh[(1+i)\ell/\delta_s]}. \end{aligned} \tag{VI.19}$$

Similarly, TX...s use cylindrical geometry in computing f_κ, f_ν, and ϵ_s:

$$\begin{aligned} f_\kappa &= \frac{2J_1[(i-1)r_0/\delta_\kappa]}{(i-1)(r_0/\delta_\kappa)J_0[(i-1)r_0/\delta_\kappa]}, \quad f_\nu = \frac{2J_1[(i-1)r_0/\delta_\kappa]}{(i-1)(r_0/\delta_\kappa)J_0[(i-1)r_0/\delta_\kappa]}, \\ \epsilon_s &= \left(\frac{K\rho_m c_p}{K_s\rho_s c_s}\right)^{1/2} \frac{f_\kappa(1+i)r_0/2\delta_\kappa}{\tanh[(1+i)\ell/\delta_s]}. \end{aligned} \tag{VI.20}$$

In TX..., the radius is that of one circular pore, so that for a heat exchanger comprised of N circular pores, the total cross-sectional area available to the working fluid is $N\pi r_0^2 =$ (Area)(GasA/A).

In SX...,

$$\frac{dp_1}{dx} = -i\omega\rho_{\rm m}\left[1+\frac{(1-\phi)^2}{2(2\phi-1)}\right]\langle u_1\rangle - \frac{\mu}{r_{\rm h}^2}\left(\frac{c_1(\phi)}{8}+\frac{c_2(\phi)R_1}{3\pi}\right)\langle u_1\rangle, \tag{VI.21}$$

$$\frac{d\langle u_1\rangle}{dx} = -\frac{i\omega\gamma}{\rho_{\rm m}a^2}p_1 + \frac{i\omega T_{\rm m}\beta^2}{\rho_{\rm m}c_p}\frac{\epsilon_{\rm s}+(g_c+e^{2i\theta_p}g_v)\epsilon_h}{1+\epsilon_{\rm s}+(g_c+e^{2i\theta_T}g_v)\epsilon_h}p_1, \tag{VI.22}$$

using

$$c_1(\phi) = 1268 - 3545\phi + 2544\phi^2, \quad c_2(\phi) = -2.82 + 10.7\phi - 8.6\phi^2, \tag{VI.23}$$

$$b(\phi) = 3.81 - 11.29\phi + 9.47\phi^2, \tag{VI.24}$$

$$R_1 = 4\left|\langle u_1\rangle\right| r_{\rm h}\rho_{\rm m}/\mu, \tag{VI.25}$$

$$\epsilon_{\rm s} = \phi\rho_{\rm m}c_p/(1-\phi)\rho_{\rm s}c_{\rm s}, \quad \epsilon_h = 8ir_{\rm h}^2/b(\phi)\sigma^{1/3}\delta_\kappa^2, \tag{VI.26}$$

$$\delta_\kappa^2 = 2K/\omega\rho_{\rm m}c_p, \tag{VI.27}$$

$$\theta_p = \text{phase}(\langle u_1\rangle) - \text{phase}\left(p_1\right), \quad \theta_T = \text{phase}(\langle u_1\rangle) - \text{phase}\left(\langle T\rangle_{u,1}\right), \tag{VI.28}$$

$$g_c = \frac{2}{\pi}\int_0^{\pi/2}\frac{dz}{1+R_1^{3/5}\cos^{3/5}(z)}, \quad g_v = -\frac{2}{\pi}\int_0^{\pi/2}\frac{\cos(2z)\,dz}{1+R_1^{3/5}\cos^{3/5}(z)}. \tag{VI.29}$$

Here, the spatial average oscillatory velocity $\langle u_1\rangle = \langle U_1\rangle/\phi A$, where ϕ is volumetric porosity and A is regenerator cross sectional area. These expressions were derived with the assumption that the thermal and viscous penetration depths are much larger than r_h.

In HX...s and TX...s, metal temperature is computed relative to fluid mean temperature using

$$\Delta T = \frac{\dot{Q}}{K}\frac{y_{\rm eff}}{\Pi x_{\rm eff}} \tag{VI.30}$$

where

$$\begin{aligned} x_{\rm eff} &= \min\{\text{peak-to-peak displacement amplitude, HX length}\} \\ y_{\rm eff} &= \min\{\delta_\kappa, r_{\rm H}\}, \end{aligned}$$

with hydraulic radius r_H equal to y_0 for `HX...`s and equal to half the circular pore radius for `TX...`s. This expression may be quite inaccurate, but we believe it is better than nothing. A little experimental evidence for it is presented in *J. Acoust. Soc. Am.* **92**, 1151 (1992). It is the only computation in DELTAE that is not correct in the acoustic approximation. If you dislike it, use the gas temperatures (available as outputs in the stack segment) instead of the metal temperatures for plotting or targeting (using freetargets). In `SX...`s, the metal temperature is computed relative to fluid mean temperature using

$$\Delta T = \frac{\dot{Q}}{K} \frac{r_h^2 (g_c - g_v)}{b(\phi)\phi A x_{eff}} \tag{VI.31}$$

where again $x_{eff} = \min\{\text{peak-to-peak displacement amplitude, HX length}\}$.

There are 3 kinds of heat exchanger, depending on position relative to stack or stacks: `HXFRSt`, `HXLASt`, and `HXMIDl` (and similarly for `SX...` and `TX...`. For '`FRSt` and '`MIDl`, the heat flow $\dot{Q}$ is an input for each pass of DELTAE; for '`LASt` it is a result. Positive heat flows into the apparatus.

'`FRSt` or '`MIDl`: $\dot{H}_{out} = \dot{H}_{in} + \dot{Q}$.

'`LASt`: $\dot{Q} = \dot{H}_{out} - \dot{H}_{in}$. $\dot{H}_{out}$ = [0 if next segment is `INSDUct` or `INSCOne`; $\dot{W}_{out}$ otherwise].

B.5 Stacks

Segment types: `STKSLab, STKREct, STKCIrc, STKDUct, STKCOne, STKPIns, STKSCreen, STKPOwerlaw`

Sample input-file segments:

```
STKSLAB   parallel-plate stack
SAMEAS 1  Area
0.724     GasA/A
7.85e-2 m Length
1.8e-4 m  y0 (half the plate spacing)
4.0e-5 m  Lplate
SAMEAS 0  Gas
kapton    Solid

STKRECT  rectangular-pore stack
SAMEAS 1  Area
0.694     GasA/A
7.85e-2 m Length
2.0e-4 m   a (half of pore width)
```

```
4.0e-5 m  Lplate
4.0e-4 m   b  (pore area is 2a times 2b)
SAMEAS 0  Gas
stainless Solid


STKCIRC approximates hexagonal honeycomb stack
SAMEAS 1  (m^2) total area
0.81     gas area/total area
0.279    (m) length
0.50e-3  (m) radius of circular pore
0.05e-3  (m) L:half of sht thcknss
helium    gas type
stainless stack material

STKDUCT  boundary-layer approx
0.01 m2 area of gas
0.4  m perimeter (this duct is square)
1. m length
0.001 m2 wall material's cross-sectional area
helium
stainless

STKCONE  boundary-layer w/ taper
0.01  m2 area of gas
0.35  m  perimeter
1. m length
sameas 8a
sameas 8b
0.001 m2 wall material's cross-sectional area
helium
stainless

STKPINS  Muller/Keolian pinstack invention
sameas 2a a area   m^2
3.2e-4    b 2y0     m   2y0 = nearest-neighbor center-to-center distance
!                           in the hexagonal lattice
0.1       c Length  m
4.e-5     d R pin   m   pin radius
helium
stainless

 STKSCreen      a screen regenerator
 sameas  1a  a Area      m^2  cross section of regenerator
    .673     b VolPor         volumetric porosity
   5.500E-02 c Length     m
   1.830E-05 d   r_H      m   hydraulic radius
    .300     e KsFrac         fudge factor F for solid conduction
 sameas  0  Gas type
 stainless  Solid type

 STKPOwerlaw      an etched foil regenerator
 sameas  1a  a Area      m^2  cross section of regenerator
    .700     b VolPor         volumetric porosity
    0.04     c Length     m
   40.e-6    d   r_H      m   hydraulic radius
    .300     e KsFrac         fudge factor for solid conduction
   36.       f f_con
```

```
  1.0       g f_exp
  24.       h h_con
  0.8       i h_exp
sameas  0  Gas type
stainless  Solid type
```

Use:

If you don't know what stacks are used for, read some background material on thermoacoustics.

Use `STKSLab` for parallel-plate or jellyroll stacks (or regenerators). Use `STKREct` for square or rectangular pores whose aspect ratio is not large [see Arnott, Bass, & Raspet, *J. Acoust. Soc. Am.* **90**, 3228 (1991).]. Use `STKCIrc` for circular or hexagonal pores. Use STKPIns for stacks comprised of pin arrays (see *J. Acoust. Soc. Am.* **94**, 941 (1993)). If pore size or plate separation is much greater than thermal and viscous penetration depths, use `STKDUct` or `STKCOne`. Use `STKSCreen` for stacked-screen regenerator (see Swift and Ward, "Simple harmonic analysis of stacked-screen regenerators," submitted to *J. Thermophys. and Heat Trans.* (1995)). Use `STKPOwerlaw` for Ron Yaron's etched-foil regenerator, or any other regenerator for which friction factor and heat-transfer coefficients follow power laws in Reynolds number.

Each end of a stack must abut a heat exchanger or another stack.

Computation algorithm:

Except in `STKSCreen` and `STKPOwerlaw`, pressure propagates according to Rott's wave equation

$$\left(1+\frac{(\gamma-1)f_\kappa}{1+\epsilon_s}\right)p_1+\frac{\rho_m a^2}{\omega^2 A_{\rm fluid}}\frac{d}{dx}\left(A_{\rm fluid}\frac{1-f_\nu}{\rho_m}\frac{dp_1}{dx}\right)-\beta\frac{a^2}{\omega^2}\frac{(f_\kappa-f_\nu)}{(1-\sigma)(1+\epsilon_s)}\frac{dT_m}{dx}\frac{dp_1}{dx}=0, \tag{VI.32}$$

subject to the condition that energy flow $\dot{H}_2$ is independent of x, which imposes the following condition on $T_m(x)$:

$$\begin{aligned}\dot{H}_2 &= \frac{A_{\rm fluid}}{2\omega\rho_m}\Im\left[\frac{d\tilde{p}_1}{dx}p_1(1-\tilde{f}_\nu-\frac{T_m\beta(f_\kappa-\tilde{f}_\nu)}{(1+\epsilon_s)(1+\sigma)})\right]\\ &+ \frac{A_{\rm fluid}c_p}{2\omega^3\rho_m(1-\sigma)}\frac{dT_m}{dx}\frac{dp_1}{dx}\frac{d\tilde{p}_1}{dx}\Im\left[\tilde{f}_\nu+\frac{(f_\kappa-\tilde{f}_\nu)(1+\epsilon_s f_\nu/f_\kappa)}{(1+\epsilon_s)(1+\sigma)}\right]\end{aligned}$$

$$- \quad (A_{\text{fluid}}K + A_{\text{solid}}K_s)\frac{dT_m}{dx} \tag{VI.33}$$

For `STKSLab`,

$$\begin{aligned} f_\kappa &= \frac{\tanh[(1+i)y_0/\delta_\kappa]}{(1+i)y_0/\delta_\kappa}, \quad f_\nu = \frac{\tanh[(1+i)y_0/\delta_\nu]}{(1+i)y_0/\delta_\nu}, \\ \epsilon_s &= \left(\frac{K\rho_m c_p}{K_s\rho_s c_s}\right)^{1/2} \frac{\tanh[(1+i)y_0/\delta_\kappa]}{\tanh[(1+i)\ell/\delta_s]}. \end{aligned} \tag{VI.34}$$

For `STKREct`,

$$\begin{aligned} f_\kappa &= 1 - \frac{64}{\pi^4}\sum_{\substack{m,n \\ \text{odd}}} \frac{1}{m^2n^2Y_{mn}(\delta_\kappa)}, \quad f_\nu = 1 - \frac{64}{\pi^4}\sum_{\substack{m,n \\ \text{odd}}} \frac{1}{m^2n^2Y_{mn}(\delta_\nu)}, \\ \epsilon_s &= \left(\frac{K\rho_m c_p}{K_s\rho_s c_s}\right)^{1/2} \frac{f_\kappa(1+i)ab/\delta_\kappa(a+b)}{\tanh[(1+i)\ell/\delta_s]}, \\ \text{where} \qquad Y_{mn}(\delta) &= 1 - i\frac{\pi^2\delta^2}{8a^2}b^2(b^2m^2 + a^2n^2) \end{aligned} \tag{VI.35}$$

For `STKCIrc`,

$$\begin{aligned} f_\kappa &= \frac{2J_1[(i-1)r_0/\delta_\kappa]}{(i-1)(r_0/\delta_\kappa)J_0[(i-1)r_0/\delta_\kappa]}, \quad f_\nu = \frac{2J_1[(i-1)r_0/\delta_\kappa]}{(i-1)(r_0/\delta_\kappa)J_0[(i-1)r_0/\delta_\kappa]}, \\ \epsilon_s &= \left(\frac{K\rho_m c_p}{K_s\rho_s c_s}\right)^{1/2} \frac{f_\kappa(1+i)r_0/2\delta_\kappa}{\tanh[(1+i)\ell/\delta_s]}. \end{aligned} \tag{VI.36}$$

For `STKDUct` or `STKCOne`,

$$\begin{aligned} f_\kappa &= (1-i)\Pi\delta_\kappa/2A, \quad f_\nu = (1-i)\Pi\delta_\nu/2A, \\ \epsilon_s &= \left(\frac{K\rho_m c_p}{K_s\rho_s c_s}\right)^{1/2} \frac{1}{\tanh[(1+i)\ell/\delta_s]} \text{ , where } \ell = \frac{\text{wall x-sect area}}{\text{perimeter}}, \end{aligned} \tag{VI.37}$$

so long as $2A/\Pi\delta_\nu < 30$. Otherwise, for $2A/\Pi\delta_\nu < 25$, the functions are the same as for `STKCIrc`. In between, a linear combination is used.

For `STKPIns`,

$$\begin{aligned} f_\nu &= -\frac{\delta_\nu}{(i-1)}\frac{2r_i}{r_o^2 - r_i^2}\frac{Y_1[(i-1)r_o/\delta_\nu]J_1[(i-1)r_i/\delta_\nu] - J_1[(i-1)r_o/\delta_\nu]Y_1[(i-1)r_i/\delta_\nu]}{Y_1[(i-1)r_o/\delta_\nu]J_0[(i-1)r_i/\delta_\nu] - J_1[(i-1)r_o/\delta_\nu]Y_0[(i-1)r_i/\delta_\nu]}, \\ f_\kappa &= -\frac{\delta_\kappa}{(i-1)}\frac{2r_i}{r_o^2 - r_i^2}\frac{Y_1[(i-1)r_o/\delta_\kappa]J_1[(i-1)r_i/\delta_\kappa] - J_1[(i-1)r_o/\delta_\kappa]Y_1[(i-1)r_i/\delta_\kappa]}{Y_1[(i-1)r_o/\delta_\kappa]J_0[(i-1)r_i/\delta_\kappa] - J_1[(i-1)r_o/\delta_\kappa]Y_0[(i-1)r_i/\delta_\kappa]}, \end{aligned}$$

and

$$\epsilon_s = \left(\frac{K\rho_m c_p}{K_s\rho_s c_s}\right)^{1/2} \frac{J_0(\sqrt{-i\omega/\kappa_s}r_i)}{J_1(\sqrt{-i\omega/\kappa_s}r_i)} f_\kappa\sqrt{-i\omega/\kappa}\frac{r_o^2 - r_i^2}{2r_i}. \tag{VI.38}$$

In STKSLabs, STKRECts, STKCIrcs, and STKPIns, the "Area" (the first line of the input file) is the total cross sectional area of the stack assembly, including both fluid cross section and solid cross section. In STKSLabs, STKRECts, and STKCIrcs, $A_{\rm fluid} = (\texttt{Area}) \times (\texttt{GasA/A})$ and $A_{\rm solid} = (\texttt{Area}) \times (\texttt{1} - \texttt{GasA/A})$. Plate thickness (the 4th line of the input file) is used only for computing ϵ_s, not for computing heat conduction along x or what fraction of the Area is available to the fluid. This allows separate accounting for area blocked by "ideal" fins and by support struts or other structure. In most cases, ϵ_s is near 0, so plate thickness need not be specified with much accuracy; GasA/A is far more important. Because of the need to compute specialized functions, STKCIrcs compute more slowly than STKSLabs or STKDUcts; STKPIns are slower still, and STKRects are very slow, especially for large aspect ratios. In the latter case, we recommend that STKSLabs be used until initial guesses and geometry are very close to finalized for this reason.

In stacked screen regenerators, pressure, volumetric velocity, mean temperature evolve according to

$$\frac{dp_1}{dx} = -i\omega\rho_{\rm m}\left[1 + \frac{(1-\phi)^2}{2(2\phi-1)}\right]\langle u_1\rangle - \frac{\mu}{r_{\rm h}^2}\left(\frac{c_1(\phi)}{8} + \frac{c_2(\phi)R_1}{3\pi}\right)\langle u_1\rangle\,, \tag{VI.39}$$

$$\begin{aligned}\frac{d\langle u_1\rangle}{dx} &= -\frac{i\omega\gamma}{\rho_{\rm m}a^2}p_1 + \beta\frac{dT_{\rm m}}{dx}\langle u_1\rangle + \\ &\quad i\omega\beta\left[\frac{T_{\rm m}\beta}{\rho_{\rm m}c_p}\frac{\epsilon_{\rm s} + (g_c + e^{2i\theta_p}g_v)\epsilon_h}{1+\epsilon_{\rm s} + (g_c + e^{2i\theta_T}g_v)\epsilon_h}p_1 - \frac{1}{i\omega}\frac{dT_{\rm m}}{dx}\frac{\epsilon_{\rm s} + (g_c - g_v)\epsilon_h}{1+\epsilon_{\rm s} + (g_c + e^{2i\theta_T}g_v)\epsilon_h}\langle u_1\rangle\right],\end{aligned} \tag{VI.40}$$

$$\begin{aligned}\frac{dT_{\rm m}}{dx} &= \left\{\Re\left[\left(T_{\rm m}\beta\frac{\epsilon_{\rm s} + \epsilon_h(g_c + e^{2i\theta_p}g_v)}{1+\epsilon_{\rm s} + \epsilon_h(g_c + e^{2i\theta_T}g_v)} + 1 - T_{\rm m}\beta\right)p_1\widetilde{\langle u_1\rangle}\right] - \frac{2\overline{H_2}}{\phi A}\right\} \\ &\quad \Big/\left\{\frac{\rho_{\rm m}c_p}{\omega}\Im\left[\frac{\epsilon_{\rm s} + \epsilon_h(g_c - g_v)}{1+\epsilon_{\rm s} + \epsilon_h(g_c + e^{2i\theta_T}g_v)}\right]\langle u_1\rangle\,\widetilde{\langle u_1\rangle} + 2K_{\rm eff}\frac{1-\phi}{\phi}\right\},\end{aligned} \tag{VI.41}$$

using

$$c_1(\phi) = 1268 - 3545\phi + 2544\phi^2, \quad c_2(\phi) = -2.82 + 10.7\phi - 8.6\phi^2, \tag{VI.42}$$

$$b(\phi) = 3.81 - 11.29\phi + 9.47\phi^2, \tag{VI.43}$$

$$R_1 = 4\,|\langle u_1\rangle|\,r_{\rm h}\rho_{\rm m}/\mu, \tag{VI.44}$$

$$\epsilon_{\rm s} = \phi\rho_{\rm m}c_p/(1-\phi)\rho_{\rm s}c_{\rm s}, \quad \epsilon_h = 8ir_{\rm h}^2/b(\phi)\sigma^{1/3}\delta_\kappa^2, \tag{VI.45}$$

$$\delta_\kappa^2 = 2K/\omega\rho_{\rm m}c_p, \tag{VI.46}$$

$$\theta_p = \text{phase}(\langle \text{u}_1\rangle) - \text{phase}\,(\text{p}_1)\,, \quad \theta_{\rm T} = \text{phase}(\langle \text{u}_1\rangle) - \text{phase}\left(\langle \text{T}\rangle_{\text{u,1}}\right), \tag{VI.47}$$

$$g_c = \frac{2}{\pi}\int_0^{\pi/2} \frac{dz}{1+R_1^{3/5}\cos^{3/5}(z)}, \quad g_v = -\frac{2}{\pi}\int_0^{\pi/2} \frac{\cos(2z)\,dz}{1+R_1^{3/5}\cos^{3/5}(z)}. \tag{VI.48}$$

Here, the spatial average oscillatory velocity $\langle u_1\rangle = \langle U_1\rangle / \phi A$, where ϕ is volumetric porosity and A is regenerator cross sectional area; and $K_{\text{eff}} = FK_s$ where F is a fudge factor to reduce thermal conduction along x due to the poor thermal contact between adjacent screen layers (Radebaugh recommends $F \leq 0.3$). These expressions were derived with the assumption that viscous and thermal penetration depths are much larger than r_h.

`STKPOwerlaw` segments are calculated in the same manner as `STKSCRN`'s, with a few exceptions. The friction factor and heat transfer coefficients are given by

$$\begin{aligned} f &= f_{con}R^{f_{exp}}, \\ St\,Pr^{2/3} &= h_{con}R^{h_{exp}}, \end{aligned}$$

where Reynolds number R is defined in the usual way as

$$R = \frac{4U_1 r_h \rho}{\phi A \mu}.$$

[Note: this is Fanning friction factor, the friction factor used by Kays and London, so that instantaneously $dp/dx = (f/r_h)\,\frac{1}{2}\rho u^2$.] The pressure equation is replaced by

$$\frac{dp_1}{dx} = -i\omega\rho_{\text{m}}\left[1+\frac{(1-\phi)^2}{2(2\phi-1)}\right]\langle u_1\rangle - I_f\frac{\mu}{8r_{\text{h}}^2}f_{\text{con}}R_1^{1-f_{\text{exp}}}\langle u_1\rangle \tag{VI.49}$$

where

$$I_f = \frac{2}{\pi}\int_0^{\pi}\sin^{3-f_{\text{exp}}}(z)dz \tag{VI.50}$$

In the volumetric velocity and mean temperature equations, these parameters are redefined for the power law stack:

$$g_c = R_1^{h_{\text{exp}}-1}\frac{2}{\pi}\int_0^{\pi/2}\cos^{h_{\text{exp}}-1}(z)dz \tag{VI.51}$$

$$\begin{aligned} g_v &= -R_1^{h_{\text{exp}}-1}\frac{2}{\pi}\int_0^{\pi/2}\cos 2z\cos^{h_{\text{exp}}-1}(z)dz \\ b(\phi) &= h_{\text{con}}. \end{aligned} \tag{VI.52}$$

Values of $\phi, r_{\text{h}}, F = K_{\text{eff}}/K_s, f_{\text{con}}, f_{\text{exp}}, h_{\text{con}}$, and h_{extp} for particular etched foil regenerators can be obtained from Ran Yaron.

In both `STKSCreen` and `STKPOwerlaw` segments, the trigonometric integrals are not evaluated by DELTAE; these integrals were performed once, off-line. We now use simple functional fits during computation of either segment type.

B.6 Begin, ends, mean-flow mode

Segment types: `TITLE, BEGIN, HARDEnd, SOFTEnd, MEANFLOW`

Sample input-file segments:

```
TITLE   comments here are reproduced in .DAT and .OUT

BEGIN
1.0e6 Pa    Mean P
500. Hz     Freq.
300. K      T-beg
3.0e4 Pa    |p|@0
0.0 deg     Ph(p)0
5.0e-4 m3/s |V|@0
0.000 deg   Ph(V)0
helium      Gas

MEANFLOW
1.E-04     U_m    m^3/s
sameas  0  Gas type

HARDEND
0.000     R(1/Z)
0.000     I(1/Z)
SAMEAS 0  Gas type

SOFTEND
0.  Re(Z)
0.  Im(Z)
water
```

Use:

The initial segments of all input files must be `TITLE` and `BEGIN`. `TITLE` is just used to give a comment field that gets reproduced in all subsequent files, so put a descriptive name in its comment field. `BEGIN` is counted as the zeroth segment of the file. It is used to initialize variables that are the same in all segments (i.e., frequency and mean pressure), and those five variables required each pass of DeltaE to get started (*i.e.*, real and imaginary parts of pressure amplitude and volume velocity, and mean temperature). (Gas type isn't really used here, but you have to give one anyway.)

`MEANFlow`, when used, should always be in segment 1, immediately following the `BEGIN` statement. Its presence establishes a constant mean mass flux through the subsequent segments, and modifies the behavior of mean-flow savvy segments (currently, these are:

HXFRST, HXMIDL, HXLAST, STKSLAB, STKRECT, and STKCIRC). This feature is still very experimental.

Often, the final segment (except free targets) will be either HARDEnd or SOFTEnd. These contain two default targets. Use HARDEnd if you want the complex volume velocity at the end of the apparatus to be zero. This is the usual case in a closed system. Use SOFTEnd if you want complex pressure amplitude at the end to be zero. We find this useful for symmetrical systems, where SOFTEnd indicates that the rest of the apparatus is a mirror image of what is in the input file, and forces a complex pressure node. In both 'ENDs, the complex impedances are made dimensionless according to $Z = Ap_1/\rho a U_1$, where A is the area of the last segment with an area, and ρ and a are evaluated at the local temperature.

Disable these as targets if you want DeltaE to ignore the impedance. This approach is useful in early stages of debugging a new model that doesn't readily converge—it may let you see what's out of whack. Set these targets nonzero to model a nonzero end impedance—or use BRANCh or OPNBRanch.

B.7 Free targets

Segment types: FREETarget, DIFFTarget, PRODTarget, QUOTArget, EFFRTarget, COPRTarget, VOLMTarget, CONSTants

Sample input-file segments:

```
FREETARGET
500.   Watts of power targeted at driver
3G     Address of computed power at driver

DIFFTARGET
0.00    a targeted difference
1B      b D1Addr
1L      c D2Addr

PRODTARGET similar to DIFFTarget.
0.00    a targeted product
1B      b M1Addr
1L      c M2Addr

QUOTARGET
1.0  desired quotient
1A   numerator address
6A   denominator address

EFFRTARGET
0.2  desired 2nd law efficiency
```

```
7F   work (numerator address)
4G   heat (denominator address)
4H   T hot address
6H   T cold address

COPRTARGET
0.2  desired 2nd law efficiency
7G   heat (numerator address)
2F   work (denominator address)
6H   T hot address
4G   T cold address

VOLMTARGET
0.50    a targeted volume (cubic meters)
1A      b BegAddr
10A     c EndAddr

CONST      test of CONST
1.00     a    So                      2.250E+03 A So*PLo
2.00     b    Si                      0.000     B Si*PLi
3.00     c   C_1                       3.00     C   C_1
4.00     d   C_2                       4.00     D   C_2
5.00     e   C_3                       5.00     E   C_3
6.00     f   C_4                       6.00     F   C_4
7.00     g   C_5                       7.00     G   C_5
8.00     h   C_6                       8.00     H   C_6
9.00     i   C_7                       9.00     I   C_7
10.0     j   C_8                       10.0     J   C_8
11.0     k   C_9                       11.0     K   C_9
12.0     l   C_10                      12.0     L   C_10
helium     Gas type
ideal      Solid type
```

Use:

Use this class of segments to create targets other than DeltaE default targets (which include only end impedances and heat exchanger heats and temperatures). You may also use them for simple arithmetic operations on results. While `CONSTants` is not itself a free target segment, its use is usually in conjunction with them. Also, it *does* require a fluid and solid line (which are ignored). Free targets do not expect such lines.

Computation algorithms:

`FREETarget:` no computation.

`DIFFTarget:` result = `[D1Addr]` − `[D2Addr]`, where `[]` signifies value calculated at this address.

`PRODTarget:` result = `[M1Addr]` × `[M2Addr]`.

`QUOTArget:` result = `[NumAdr]` / `[DenAdr]`.

`EFFRTarget:` result = $\dfrac{W}{Q_h}\dfrac{T_h}{T_h - T_c}$

`COPRTarget:` result = $\dfrac{Q_c}{W}\dfrac{T_h - T_c}{T_c}$

`VOLMTarget:` result = sum of the volumes in all duct, cone, stack, compliance, and heat exchanger segments beginning with `BegAddr` and ending with `EndAddr` (parameter letters are inconsequential). Porosity is not used in calculating this volume—that is, porosity is always effectively 100%.

`CONSTants:` output = input, except for outputs `A` and `B`, where
`A` = `a`× current outer plot loop independent variable;
`B` = `b`× current inner plot loop independent variable.
Since these constants are now *outputs*, their addresses can be used with any of the free targets listed above.

B.8 Tees and unions

Segment types: `TEE, TBRANch, UNION, HBRANch, HUNIOn`

Sample input-file segments:

```
TEE        branch file to load
branch.in

TBRAN      the fork
 4.412E+07 a Re(Zb) Pa-s/m^3  G
-3.528E+06 b Im(Zb) Pa-s/m^3  G
sameas  0  Gas type
ideal      Solid type

UNION      below the branch
     4    segment number of SOFTEND of the TBRANCH
    3.e4  |p| @ end (Pa)
     0.   ph(p) @end
sameas 0  Gas type
ideal      Solid type

HBRAN      fork with Hfrac
 4.412E+07 a Re(Zb) Pa-s/m^3  G
-3.528E+06 b Im(Zb) Pa-s/m^3  G
 0.49      c  Hfrac           G
sameas  0  Gas type
ideal      Solid type
```

```
HUNION    H matching joint      5
     10    segment number of SOFTEND of the HBRANCH
    4.e3  |p|End  Pa    =5A?
     0.   Ph(p)E        =5B?
   300.    T-est  K     =5G?
sameas 0  Gas type
ideal      Solid type
```

Use:

Use `TBRANch` for branched systems too complicated for `BRANCh` or `OPNBRanch`.

When it encounters a `TBRANch`, DeltaE treats subsequent segments as the sequential members of a branch until it reaches a `HARD-` or `SOFTEnd`, then it "returns to the trunk," treating the rest of the segments as trunk members. If the system is multiply connected, a `UNION` segment in the trunk tells DeltaE where to connect the branch's `SOFTEnd` back to the trunk.

If `UNION` is used, the branch's `SOFTEnd` impedance targets should not be used; instead, enable the `UNION`'s targets to ensure that (complex) p is equal at the `SOFTEnd` of the branch and at the `UNION` in the trunk. The guessed branch impedance determines how the (complex) volume velocity splits up at the `TBRANch`; volume velocities add at a `UNION`. `UNION` targets are a special case in that their input values are dynamically rewritten by DeltaE during iterations, depending on the most recent results at the named `SOFTEnd`. The real input parameters (magnitude and phase of pressure) can have any value when the input file is written. DeltaE will overwrite them during each pass with the current magnitude and phase of pressure at the referenced `SOFTEnd`.

`BRANCH` and `UNION` are intended for duct networks, where temperature is constant and hence p_1 and U_1 are the variables of interest. For more complex systems, the segments `HBRANCH` and `HUNION` are energy-conserving versions of `BRANCH` and `UNION`. Use them if you are branching at locations where $\dot{H}_2 \neq \dot{W}$, such as at a branch to a second stage regenerator within a two-stage pulse tube refrigerator. `HBRANCH` incorporates a potential guess `Hfrac`, giving the fraction of the incoming energy that goes into the branch. Use `Hfrac` as a guess to hit a target down the branch, such as a temperature. `HUNION` incorporates an additional potential target, that the temperature in the trunk at the union be equal to that at the associated branch end. energy flow.

When DeltaE encounters a `TEE`, it loads the named file into the model, and replaces the `BEGIN` segment of the branch file with a `TBRANch` segment. It tries to guess starting values for the complex branch impedance, and then adjusts the addresses in any `sameas`

declarations and free target-type segments occurring in the branch (or after the branch point) by the number of segments in the branch. Once the file has been read in, the `TEE` segment disappears—the `.out` file and (d)isplayed segments will be the composite model. The file may have any name (*e.g.* `branch.in, stub.out, branch.tee`), but it must be specified with the complete suffix.

Computation algorithm:

At a `TBRANch`, the branch complex impedance determines how much volume velocity leaves the trunk into the branch. At a `UNION`, exit volume velocity equals inlet volume velocity plus volume velocity at the branch's `SOFTEnd`.

B.9 Acoustical decomposition

Segment type: `DECOMpose`

Sample input-file segment:

```
DECOMp     Termination
  8.100E-03 a Area      m^2              15.8     A |Pin|     Pa
                                         31.9     B |Pref|    Pa
sameas  0  Gas type                      4.11     C RflCoe   W/W
ideal      Solid type                   -77.9     D PhI-R    deg
```

Use:

Use `DECOM` to decompose the acoustic field into *incident* and *reflected* pressure waves; that is, solve for $P_{\text{in}}, P_{\text{ref}}$, and $\phi_I - \phi_R$ in the equation

$$p_1 = P_{\text{in}} e^{i(-kx+\phi_I)} + P_{\text{ref}} e^{i(kx+\phi_R)}, \tag{VI.53}$$

where $P_{\text{in}}, P_{\text{ref}}$, and k are considered real for this segment.

Computation algorithm:

Since the segments surrounding the `DECOM` segment are generally lossy in DELTAE, its results are strictly valid only at that point. The magnitudes are calculated from

$$\begin{aligned} P_{\text{in}} &= \frac{|p_1+U_1\rho a/A|}{2} \\ P_{\text{ref}} &= \frac{|p_1-U_1\rho a/A|}{2} \end{aligned} \tag{VI.54}$$

and the phase difference is given by

$$\text{phase}\left(\frac{p_1+U_1\rho a/A}{p_1-U_1\rho a/A}\right) \tag{VI.55}$$

The sound power reflection coefficient, $(P_{\text{ref}}/P_{\text{in}})^2$, is also found and given as output `C`.

B.10 Thermophysical properties dump

Segment type: `THERMOphys`

Sample input-file segment:

```
THERMO
sameas 0
```

Use:

Use `THERMO` to provide a record of thermophysical properties and penetration depths at a given location in the apparatus. With plotting features, can be used to generate a table of thermophysical properties.

B.11 ALPHABETICAL LISTING AND CROSS-REFERENCE

`BEGIN`: **(B.6)** Initializes p_1, U_1, and T_m at the beginning, and sets global f, p_m.

`BRANCH`: **(B.3)** A side-branch with frequency-independent complex impedance.

`COMPLIANCE`: **(B.2)** A lumped acoustic compliance (with surface losses).

`CONSTANTS`: **(B.7)** Allows constants and plot independent variables to be used in `FREETARGETS`.

`COPRTARGET`: **(B.7)** Allows targeting of ratio of refrigerator COP to Carnot's COP.

`DECOMPOSE`: **(B.9)** Decomposes wave into forward and backward traveling components.

`DIFFTARGET`: **(B.7)** Allows targeting of difference of two results.

`ENDCAP`: **(B.2)** A surface area with $|p_1|^2 \delta_\kappa$ loss.

`EFFRTARGET`: **(B.7)** Allows targeting of ratio of engine efficiency to Carnot's efficiency.

`FREETARGET`: **(B.7)** Allows use of non-default target.

`HARDEND`: **(B.6)** Default inverse-impedance targets, for hard model termination.

`HBRANCH`: **(B.8)** An energy-conserving `BRANCH` for multi-stage refrigerators.

`HXFRST`: **(B.4)** A parallel-plate heat exchanger before a stack.

`HXLAST`: **(B.4)** A parallel-plate heat exchanger after a stack.

`HXMIDL`: **(B.4)** A parallel-plate heat exchanger between two stacks.

`HUNION`: **(B.8)** An energy-summing, temperature-matching `UNION`.

`IDUCER`: **(B.3)** A current-driven transducer attached as a side branch (and independent of frequency).

`IEDUCER`: **(B.3)** An enclosed (i.e. series) current-driven transducer (and independent of frequency).

`IESPEAKER`: **(B.3)** An enclosed (i.e. series) current-driven electrodynamic transducer.

`IMPEDANCE`: **(B.2)** A lumped-parameter series acoustic impedance.

`INSCONE`: **(B.1)** An insulated cone, with viscous and thermal dissipation.

`INSDUCT`: **(B.1)** An insulated duct, with viscous and thermal dissipation.

`ISOCONE`: **(B.1)** An isothermal cone, with viscous and thermal dissipation.

`ISODUCT`: **(B.1)** An isothermal duct, with viscous and thermal dissipation.

`ISPEAKER`: **(B.3)** A current-driven electrodynamic transducer, attached as a side branch.

`MEANFLOW`: **(B.6)** Enables nonzero mean flow superimposed on the acoustics.

`OPNBRANCH`: **(B.3)** A side-branch impedance with frequency dependence of 4π open radiation impedance.

PISTBRANCH: **(B.3)** A side-branch impedance with frequency dependence of baffled piston.

PRODTARGET: **(B.7)** Allows targeting of product of two results.

QUOTARGET: **(B.7)** Allows targeting of quotient of two results.

SOFTEND: **(B.6)** Default impedance targets, for mirror-image model termination; also for connection of sidebranch to UNION.

STKCIRCLE: **(B.5)** Thermoacoustic stack (or regenerator) comprised of array of circular pores.

STKCONE: **(B.5)** Thermoacoustic element comprised of a single, conical pore.

STKDUCT: **(B.5)** Thermoacoustic element comprised of a single, straight pore.

STKPINS: **(B.5)** Thermoacoustic stack (or regenerator) comprised of array of pins.

STKPOWERLAW: **(B.5)** Regenerator with friction factor and heat transfer as power laws in Reynolds number.

STKRECT: **(B.5)** Thermoacoustic stack (or regenerator) comprised of array of rectangular pores.

STKSCREEN: **(B.5)** Regenerator comprised of stacked screens.

STKSLAB: **(B.5)** Slab-geometry stack or regenerator, comprised of parallel plates.

SXFRST: **(B.4)** A screen heat exchanger before a stack.

SXMIDL: **(B.4)** A screen heat exchanger after a stack.

SXMIDL: **(B.4)** A screen heat exchanger between two stacks.

TBRANCH: **(B.8)** The beginning of a side-branch series of segments.

TEE: **(B.8)** A temporary segment that inserts a complete file into the model. The file's BEGIN segment becomes a TBRANCH.

THERMOPHYSICAL: **(B.10)** Displays properties of gas and solid at the local temperature.

TITLE: **(B.6)** Comment field required at start of every file.

TXFRST: **(B.4)** A tubular heat exchanger before a stack.

TXLAST: **(B.4)** A tubular heat exchanger after a stack.

TXMIDL: **(B.4)** A tubular heat exchanger between two stacks.

UNION: **(B.8)** Matches p_1 and adds U_1 at union between end of side branch and trunk.

`VDUCER`: **(B.3)** A voltage-driven transducer attached as a side branch (and independent of frequency).

`VEDUCER`: **(B.3)** An enclosed (i.e. series) voltage-driven transducer (and independent of frequency).

`VESPEAKER`: **(B.3)** An enclosed (i.e. series) voltage-driven electrodynamic transducer.

`VOLMTARGET`: **(B.7)** Allows targeting of total volume of a series of segments.

`VSPEAKER`: **(B.3)** A current-driven electrodynamic transducer, attached as a side branch.

C Fluids

We provide an artificial temperature floor of 10 Kelvin to prevent DeltaE from trying to use negative temperatures when it is really lost. Consequently, no temperature below 10 Kelvin can be used. In any case, most of the equations for the fluids are inaccurate when this limit is reached. This floor can be modified within the `(T)olerances/debugging` menu.

In what follows, `ta` is temperature in Kelvin, `t1` is temperature in Celsius.

Unless otherwise specified, properties are computed using fits to the data compiled in Touloukian's TPRC series.

DeltaE looks for a 10-character field to determine fluid type. Be sure to use plenty of trailing spaces after short fluid names like "air" to get comments like "gas-type" out of the field.

C.1 helium

Ideal gas approximation for equation of state (including sound speed and expansion coefficient) and specific heat. Transport from Touloukian:

```
k0=0.0025672*ta**0.716
mu=0.412e-6*ta**0.68014
```

C.2 #.###hear (helium-argon mixtures)

Number in the fluid name is helium fraction. Ideal gas approximation for equation of state and specific heat. Transport from Touloukian.

```
k0he=0.0025672*ta**0.716
amuhe=0.412e-6*ta**0.68014
k0ar=(1.39e-4*ta**0.852-1.5e-8*(ta-300.)*(ta-300.))*(1.+2.e-8*pm)
amuar=(1.77e-7*ta**0.852-25.e-12*(ta-300.)*(ta-300.))*(1.+2.e-8*pm)
k0=x1*k0ar+x2*k0he-(k0ar+k0he)*x1*x2**1.5
mu=x1*amuar+x2*amuhe+0.2*(amuar+amuhe)*x1*x2
```

C.3 #.###hexe (helium-xenon mixtures)

Number in the fluid name is helium fraction. Ideal gas approximation for equation of state and specific heat. Our fits to Touloukian's transport data are only accurate for `frxe` < 0.5 or for `frxe` $= 1.000$:

```
k0he=0.0025672*ta**0.716
amuhe=0.412e-6*ta**0.68014
k0xe=4.75e-5*ta**0.84*(1.+1.e-7*pm)
amuxe=0.187e-6*ta**0.85*(1.+25.e-9*pm)
frxe=1.-fhe
k0=k0he*fhe+k0xe*frxe-2.*(k0he+k0xe)*frxe*fhe*fhe
mu=amuhe*fhe+amuxe*frxe+(amuhe+amuxe)*frxe*fhe*fhe*(0.8+3.7*fhe*fhe*(0.25-f
rxe))
```

C.4 neon

Ideal gas approximation for equation of state and specific heat. Transport from Touloukian:

```
k0=0.001149*ta**0.65907
mu=0.735e-6*ta**0.66065
```

C.5 air

Ideal gas approximation for equation of state and specific heat. Transport from Pierce, Acoustics:

```
parameter (tps=110.4,tpa=245.4,tpb=27.6,tp0=300.,tpexp=223.8306)
k0=2.624e-2*(ta/tp0)**1.5*(tp0+tpexp)/(ta+tpa*exp (-tpb/ta))
mu=1.846e-5*(ta/tp0)**1.5*(tp0+tps)/(ta+tps)
```

C.6 nitrogen

Ideal gas approximation for equation of state and specific heat. Transport from Touloukian:

```
k0=0.0003609*ta**0.7512
mu=0.3577e-6*ta**0.6885
```

C.7 hydrogen

Ideal gas approximation for equation of state and specific heat. Transport from Touloukian:

```
k0=0.002627*ta**0.744
mu=0.19361e-6*ta**0.6723
```

C.8 deuterium

Ideal gas approximation for equation of state and specific heat. Transport from Touloukian:

```
k0=0.002795*ta**0.686
mu=0.2726e-6*ta**0.6721
```

C.9 co2 (carbon dioxide)

Ideal gas approximation for equation of state and specific heat. Transport from Touloukian:

```
k10=2.8646E-5*ta**1.1318
k20=3.692E-5*ta**1.0940
k0=k10+(pm-1.01e6)/(1.01e6)*(k20-k10)
u10=1.4187E-7*ta**.8216
u20=1.5416E-7*ta**.8094
mu=u10+(pm-1.01e6)/(1.01e6)*(u20-u10)
```

C.10 #.###nexe (neon-xenon mixtures)

Ideal gas approximation for equation of state and specific heat. Transport from ??: (Thermal conductivity not very accurate for high xenon concentrations.)

```
k0he=0.001149*ta**0.65907
amuhe=0.735e-6*ta**0.66065
k0xe=4.75e-5*ta**0.84*(1.+1.e-7*pm)
amuxe=0.187e-6*ta**0.85*(1.+25.e-9*pm)
frxe=1.-fhe(ns)
k0=k0he*fhe(ns)+k0xe*frxe-1.3*(k0he+k0xe)*frxe*fhe(ns)**2.5
mu=amuhe*fhe(ns)+amuxe*frxe+0.12*(amuhe+amuxe)*frxe*fhe(ns)**4
```

C.11 NGcbProd (natural-gas combustion products)

Natural Gas combustion products with 5% excess air. Use around 1 atm only. Data supplied by British Gas, with references to Pritchard. Molar weight is a gaussian curve fit taken from Pritchard's data between 288 K and 4000 K.

```
 gamma=1.4
 cp=gasprop(ta,1392.02d0,39.3769d0,-3.89819d0,-0.0317961d0,
&    0.0327554d0,-1.44149d-3)
 if (ta.gt.2000.) then
   mass=27.9495-7.81175*dexp(-((ta-4151.85)/1047.42)**2)
 else
   mass=27.84
 endif
 r=8314.
 a=dsqrt(gamma*r*ta/mass)
 rho=pm*mass/(r*ta)
 beta=1./ta
 k0=gasprop(ta,0.0997279d0,0.0125516d0,6.73728d-5,4.22761d-4,
&    1.43198d-4,1.35508d-5)
 mu=gasprop(ta,50.2973d0,4.68523d0,-0.12061d0,0.0140082d0,
&    -0.001488951d0,4.97968d-5)*1.d-6
 goto 900
 real*8 function gasprop(ta,a,b,c,d,e,f)
 real*8 z,ta,a,b,c,d,e,f
 z=(ta-1400)/200
 gasprop=a+z*(b+ z*(c + z*(d +z*(e+f*z))))
 return
 end
```

C.12 sodium

Data for sodium from Foust, Sodium-NaK Engineering Handbook.

```
a0=2578.
at1=-.52
ap=6.1e-7
r0=950.1
rt1=-2.2976e-1
rt2=-1.46e-5
rt3=5.638e-9
c0=1.4361e3
ct1=-5.8024e-1
ct2=4.6208e-4
k0=.918e2-4.9e-2*t1
if(t1.le.500.) then
 e1=.697
 e2=1.235e-5
else
 e1=1.04
 e2=8.51e-6
endif
a=a0+at1*t1
rho=r0+rt1*t1+rt2*t1**2+rt3*t1**3
beta=(-rt1-2.*rt2*t1-3.*rt3*t1**2)/rho
bt=beta**2-(2.*rt2+6.*rt3*t1)/rho
cp=c0+ct1*t1+ct2*t1**2
rp=1./a/a+ta*beta**2/cp
bp=-beta/(rho*a**2)+2.*at1/(rho*a**3)-beta**2/rho/cp
bp=bp-2.*ta*beta*bt/rho/cp-ta*beta**3/rho/cp
bp=bp+ta*beta**2*(ct1+2.*ct2*t1)/rho/cp/cp
cpp=-ta*(beta**2+bt)/rho
c So far, everything is evaluated at p=0.
a=a+ap*pm
rho=rho+rp*pm
beta=beta+bp*pm
cp=cp+cpp*pm
gamma=1.+ta*beta**2*a**2/cp
mu=e2*rho**(1./3.)*exp (e1*rho/ta)
```

C.13 nak-78

This is for eutectic NaK-78. Data for sodium-potassium from Foust, Sodium-NaK Engineering Handbook.

```
a0=2051.
at1=-.53
ap=0.
r0=876.4
rt1=-2.183e-1
```

```
rt2=-2.982e-5
rt3=0.
c0=970.69
ct1=-.36903
ct2=3.4309e-4
k0=21.4+2.07e-2*t1-2.2e-5*t1**2
if(t1.le.400.) then
 e1=.688
 e2=1.16e-5
else
 e1=.979
 e2=8.2e-6
endif
a=a0+at1*t1
rho=r0+rt1*t1+rt2*t1**2+rt3*t1**3
beta=(-rt1-2.*rt2*t1-3.*rt3*t1**2)/rho
bt=beta**2-(2.*rt2+6.*rt3*t1)/rho
cp=c0+ct1*t1+ct2*t1**2
rp=1./a/a+ta*beta**2/cp
bp=-beta/(rho*a**2)+2.*at1/(rho*a**3)-beta**2/rho/cp
bp=bp-2.*ta*beta*bt/rho/cp-ta*beta**3/rho/cp
bp=bp+ta*beta**2*(ct1+2.*ct2*t1)/rho/cp/cp
cpp=-ta*(beta**2+bt)/rho
c So far, everything is evaluated at p=0.
a=a+ap*pm
rho=rho+rp*pm
beta=beta+bp*pm
cp=cp+cpp*pm
gamma=1.+ta*beta**2*a**2/cp
mu=e2*rho**(1./3.)*exp (e1*rho/ta)
```

C.14 External—provided by user's file.

Files can have any name valid under the operating system under which DeltaE is running, and should end with the extension `.tpf`. If the root filename is the same as any pre-defined fluids, DeltaE will replace it's internal calculations for that fluid with those given in the user file. To request a user-defined fluid, simply use the root file name as you would any other fluid. The `.tpf` file should be in the same directory or folder as the model file. The name of the fluid is set to the root filename of the external fluid file. Up to five distinct external fluids can be used at one time.

Each property is specified by a line containing 1–10 real coefficients to be read in as C_{0-9}, where unused parameters are set to zero. The order of the property lines is ρ, c_p, K, a^2, and μ. Comment lines can be added with an initial '!', and blank lines are ignored.

Each of the five properties is derived from its 10 coefficients using the following equation:

$$\text{property} = C_0 + C_1\frac{p_m}{T + p_m C_2} + C_3 T + C_4 T^2 + C_5 T^{C_6} + C_7 p_m^2 T^{C_8} + p_m C_9, \qquad \text{(VI.56)}$$

where T and p_m are the absolute temperature (K) and mean pressure (Pa) for each point at which a segment using the fluid is evaluated.

D Solids

We provide an artificial temperature floor of 10 Kelvin to prevent DeltaE from trying to use negative temperatures when it is really lost. Consequently no temperature below 10 Kelvin can be used.

In what follows, `ta` is temperature in Kelvin, `t1` is temperature in Celsius.

DeltaE looks for a 10-character field to determine solid type. Be sure to use plenty of trailing spaces after short solid names like "`mylar`" to get comments like "`solid-type`" out of the field.

D.1 ideal

`ks, rhos`, and `cs` are effectively infinite, so $\epsilon_s = 0$.

D.2 copper

```
ks=398.-.0567*(ta-300.)
rhos=9000.
cs=420.
```

D.3 nickel

```
if (ta.lt.631) then
 ks=63.8+.08066*(631.-ta)
else
 ks=63.8+.02156*(ta-631.)
endif
rhos=8700.
cs=530.
```

D.4 stainless (stainless steel)

```
rhos=8274.55 -1055.23 *dexp(-((T1-2171.05)/2058.08)**2)
```

```
ks=(266800*ta**(-5.2)+0.21416*ta**(-1.6))**(-0.25)
cs=(1.7054e-6*ta**(-0.88962)+23324/ta**6)**(-1/3) + 15/ta
```

Prior to version 3.5b2, DELTAEś stainless steel properties were very inaccurate at cryogenic temperatures.

D.5 molybdenum

```
rhos= 10868.6 -2637.52 * exp (-((T1-11383.7)/9701.36)**2)
cs= 253.791 +0.0583812 *T1-2.73919e-06*T1**2
ks= (33.9616 -0.00947953 *T1-4.12809e-08*T1**2)*4.186
```

D.6 tungsten

```
cs=.13576e3*(1.-4805./ta**2)+.0091159*ta+2.31341e-9*ta**3
ks=135.5+1.05e4/ta-.023*ta
rhos=19254*(1.-3.*(-8.69e-5+3.83e-6*t1+7.92e-10*t1**2))
```

D.7 kapton

```
ks=0.2*(1.-exp(-ta/100.))
rhos=1445.-0.085*ta
cs=3.64*ta
```

D.8 mylar

```
ks=0.11+1.7e-4*ta
rhos=1400.-0.175*ta
cs=3.7*ta
```

D.9 External-provided by user's file.

External solids, like external fluids, are derived from coefficients in user-written text files. Up to five external solids can be used at once. Each property is specified by a line containing 1–10 real coefficients to be read in as C_{0-9}, where unused parameters are set to zero. The order of the property lines is ρ_s, c_s, and K_s. Comment lines can be added with an initial '!', and blank lines are ignored.

Each of the three properties is derived from its 10 coefficients using the following equation:

$$\text{property} = C_0 + C_1 \exp(-TC_2) + C_3 T + C_4 T^2 + C_5 T^{C_6} + C_7 p_m^2 T^{C_8} + p_m C_9. \qquad \text{(VI.57)}$$

To request a user-defined solid, simply use the root file name as you would any other solid. The `.tpf` file should be in the same directory or folder as the model file. If the name matches any pre-defined solid name, the (constant) user-defined properties will replace DeltaE's internal calculations. External solids are similar in most respects to external fluids; see Section V.C.12 for more relevant information.

E Menu Options

We list DeltaE's menu options in the order in which they appear.

`(r)un model` instructs DeltaE to begin its computation, adjusting the elements of the guess vector until either all targets are met or an error condition is reached. If one or two plot-independent variables are set, DeltaE will step through them.

`(w)rite current model state` saves the current state in a `.out` file. If the file already exists, you will be given the option of overwriting or renaming it.

`(n)ew model input file` brings a new `.in` (or `.out` file from the disk. If changes have been made in the current model, the user will be prompted to save it first.

`(R)estore vectors.` Use this option to restore all the parameters that were changed to their starting point after an unsuccesful iteration, then modify some value(s) and try again. Do not use this option if the vector table has since been edited.

If you do not respond '`y`'es to the prompt about vector restoration and you have one or both plot loops enabled, you will be give an additional option:

```
Restore to state before last (B)egin or (r)un (y|n)? n

Restore from a recently plotted point? y
```

DeltaE will now proceed to display the `.plt` file one line at a time. After each line this prompt appears:

```
Return to this state (y|n|Q)? y
```

Typing '`y`' at this point causes the independent plot variable(s) and all members of the guess vector to be returned to those values displayed in the file. Typing '`n`' (or simply <CR>) causes the next line to be displayed. '`Q`' skips to the end of the file and makes no changes. No outputs are changed when this option is executed, so the model must be `(r)un` again to update them; however, be sure to disable the outer plot loop first if you want only one point. Alternatively, you can change the step or endpoints of the plot loop and start plotting again.

This option only works on the current (open) plot file, and it is not useful until after a run which has produced plot points.

`(E)xtras` This option enters a submenu containing less commonly used features (described below).

`(d)isplay` shows information on the screen. It prompts the user to select the `.dat` file (option d), the `.plt` file (option p), the entire `out` file (option o), or a single segment in `.out`-file format (option `n`, the segment number). On PC or Unix platforms, these screens have an automatic pause feature after 23 lines are typed—press <CR> to continue, or 'q' to quit the display. There is no 'backup' yet.

`(o)utput to printer` is the same as "display" above, but for a printed copy instead of screen display.

`(f)orm feed printer` makes the printer finish the page and spit it out after doing an "output to printer."

`(t)hermophysical properties` is used to look at properties of any gas, liquid, or solid supported by DeltaE. The user is prompted to select material, temperature, pressure, etc., with current values as defaults, selectable with a carriage return. Data are displayed on the screen in this format:

```
FLUID: 0.880hexe  , 302.0 K, 20.000 bar
gamma  a(m/s) rho(kg/m^3) cp(J/kg/K) beta(1/K) k0(W/m/K) Prandtl mu(kg/s/m)
 1.67  465.91     15.356     1078.2  .3311E-02   .10586  .2604473 2.5572E-05
Frequency=   0.16 Hz, delta_nu=  1.8250E-03 m, delta_kappa=  3.5760E-03 m
Print this? (y/n):
```

`(e)xit DeltaE` returns us to the computer's operating system, prompting for several choices of saving the current model state.

`(p)lot another parameter` adds another parameter to the plot. When a number and *capital* letter is selected, its variable is added to the list of dependent plot variables. When a number and *lower case* letter is selected, its variable is used as an independent plot variable, and the user is prompted to choose it as either inner or outer loop variable, and to give its beginning, ending, and increment (or decrement) values.

`(P)lot status summary` simply displays the current plot status on the screen.

`(c)lear from vectors and plots` is used to eliminate variables from the guess vector, the target vector, the plot dependent-variable list, or the plot independent-variable list. Remember to use lower-case letters when selecting guesses,targets, or plot independent variables, and capital letters for plot dependent variables.

`(C)lear|set all guesses&targets` clears *everything* from the guess and target vectors, if they contain anything. This is most useful in the early stages of model development: If DeltaE doesn't converge, return to your initial guesses (they may be way out of line by now), clear everything, run it, and examine the results to see if one particular segment is giving ridiculous results due to a typographical error in the `.in` file. If the model has empty vectors after a previous (C)lear, selecting this option again causes DeltaE to generate a set of default iteration vectors appropriate to the model.

`(u)se in guess/result vector` allows the user to add a new variable to the guess or target vector (using a number and lower-case letter).

`(v)ector status summary` shows the current members of the guess and target vectors on the screen.

`(m)odify parameter value` followed by a segment number and lower-case line letter allows the user to change the value of a guess or input variable. Two special Uppercase pseudo-parameters are also recognized. Selecting n`G` or n`S` permits the Gas type or Solid type, respectively, of segment n to be changed by selecting interactively from a list of all defined types (including currently active user-defined properties).

`(s)pecial modes editing` allows parameter linking modes to be set for any address (segment number and parameter letter) that accepts them. Special modes allow geometric relationships to be maintained when parameters are changed by the solver, by the user, or as an independent plot loop variable.

`(D)OS command shell` temporarily suspends DeltaE and executes a new DOS command environment. This is intended to let the user examine files that are part of other models (not available under (d)isplay), to run plotting software on recent results, etc., without having to save all changes and leave DeltaE. To return to the program, type `exit`.

E.1 (E)xtra options

The following less-used menu options are accessible after entering `E` to enter the `(E)xtras` submenu:

`(S)plit segment.` This option automates the laborious process of splitting a duct segment (or anything else that has a length) into two segments, each with half the original

length, correcting the sameas and free target references, and correcting the iteration, optimization, and plot vectors. (All free targets, vectors, or sameas references to the segment specified are incremented by one; that is, the number of the original segment is incremented by one, and the 'clone' segment is effectively inserted before it.)

(K)ill segment. This option simply removes a segment from your model. It works on any type of segment (except BEGIN), and it does nothing intelligent with any lengths that are removed. The user must compensate another length where appropriate.

(I)nsert segment. DeltaE will prompt you for the correct number of parameters, giving the parameter name and units. This function is not perfectly interactive. If you make errors in typing in new parameter values, you will be left with a segment that is partly the same as the previous occupant of this spot. You may be able to recover by using the (m)odify value option in the main menu for numerical parameters. In the worst case (a bad segment type, for example), you may have to (K)ill the new segment and start over again. (I)nsert before #segments+1 is permitted to add a segment at the very end.

(G)enerate state variable plot performs a single run to generate output of position, area, temperature, pressure, and enthalpy throughout the model. The .spl file that is written contains Nprint+1 (see Tolerances/debugging, below) for each integrated segment in the model (ducts are also integrated in this mode). Non-integrated segments dsiplay one line at the beginning and one at the end processing.

(g)eometry file causes X-Y points representing a simple sketch of the model to be written to a file ending in .geo. Plotted using most any graphing software, these points will give a visual feel for the shape of the design. See the Section V.G for an example plot and discussion.

(F)lip model. This will take every segment between the BEGIN and the last HARDEND or SOFTEnd and reverse their order. Segments within TBRANches are left in their original order, however. sameas, freetarget and plot references are all adjusted and an attempt is made to reform the guess and target vectors. Each HSXFRst segment becomes an HXLAst, and *vice versa*.

(T)olerances/debugging DeltaE has numerous internal parameters that can be altered by the experienced user to control the amount of diagnostic information printed or the behavior of DeltaE's solver. The dialog that appears for this command looks like this, if we keep all the default values by hitting <CR>:

```
Nprint <= 0, save only converged endpoint to .dat file.
Nprint > 0, save and display every Nprint intermediate iterations; also
If Nprint < 0, the iteration vector line is omitted.
Nprint =  -1?

If PlotDat>=0, all error messages are announced.
```

```
 Otherwise, they are only written to the .dat file.
If PlotDat>=1, all converged endpoints are written to the .dat file.
 (for PlotDat=0, only the most recent)
PlotDat =   0?

Convergence tolerance (1.e-2 > tol > 1.5e-7 recommended):
Tolerance =    .300E-03
New value (<CR> to keep)=?

Number of Runge-Kutta steps (should be even:)
Nint =  10?

Normalization mode:  1=standard; 2=special
mode =   1?

Solver step bound factor (.01-100 recommended):
Bound =    100.
New value (<CR> to keep)=?

(Larger values of FCNerr can speed iterations, with a
slightly less accurate endpoint.  Too small a value
 can cause the solver to loose its way completely.)
Solver assumed function error (>5.e-15):
FCNerr =    .100E-09
New value (<CR> to keep)=?

Minimum Temperature (K):
Tmin=10.0
New value (<CR> to keep)=?
```

For further details of the effect of each of these parameters, refer to the discussion in Chapter V.

`(e)xit to Main` Return to the main menu. Typing `<CR>` alone has the same effect.

F Troubleshooting, Common Problems, and Suggested Techniques

The most common problem is failure of a brand-new model to converge, and the most common causes are order-of-magnitude typos in the input file and a premature attempt to run DeltaE with too many variables in the guess and target vectors. The easiest way to fix such a problem is to Clear everything from the guess and target vectors, run DeltaE, and display the `.dat` file. Often it is obvious where your typo is—one of the output variables will go wild in a supposedly innocuous segment. If not, examine the results more closely for reasonableness. Modify suspicious variables a little, to see what effect they have on results. Try to get the model close to converging on your desired targets just by modifying your desired guesses one at a time, manually. Then add one guess-target pair at a time, running DeltaE each time, examining the results, and manually modifying your other desired guess variables to try to keep your other desired target variables under

control. For further diagnostic information, try using the `Nprint` variable, found under the `(T)olerances/debugging` menu described in the previous section and in Chapter V.

It is also useful to keep the model as simple as possible. Examples:

- Rely on nonzero U in `BEGIN` instead of a transducer segment if possible.
- Don't try to model a thermoacoustically-driven thermoacoustic refrigerator from scratch without first succeeding with a thermoacoustic driver and then a piston-driven thermoacoustic refrigerator.
- Understand the acoustics of a complicated resonator before adding the stack and heat-exchanger segments.
- In really stubborn cases, start with only one segment; add segments one at a time, inspecting results carefully.

Another cause of failure to converge is poor choice of guess-vector members. Obviously, cross-sectional area of all the segments in a system has little effect on resonance frequency, but a large effect on thermoacoustic power; similarly, it has little effect on $\Im(1/Z)$ but a large effect on $\Re(1/Z)$, so don't try using area as a guess to achieve a target $\Im(1/Z)$. Clearing vector members and running DeltaE, manually modifying potential guess-vector members individually to see if they have significant effects on potential target-vector members, is often educational.

The solver within DeltaE can sometimes become stuck around a local minimum, particularly if you are making incremental changes from a model that has already converged—and often, the internal representation of the 'best guess' does not agree with what we would like for a given model. Try manually changing one of the guess vector members slightly and see if DeltaE will loose its fondness for this particular point. Or, change one of the members of the guess vector, if you can think of an appropriate alternate.

If these steps fail, consider some of the options in (T)olerances/debugging. Some pathologically difficult cases converge better with tighter tolerance, alternate normalization mode, or one of the other tuning options described at the end of Chapter V.

Always check results carefully for reasonableness, particularly when calculating complicated models or using any of DeltaE's more elaborate features. While DeltaE is a useful tool, it is far from foolproof; for example, the shooting method can easily end up generating devices that are several wavelengths long, if initial convergence is slow. All `INS`-type segments, `TBRANches` and `UNION`s containing thermoacoustic elements also deserve special skepticism.

G Error and Informational Messages

Most of DeltaE's diagnostics are meant to be self-explanatory, but some require additional information. In the following subsections, we offer some additional hints for the more obscure ones.

G.1 Convergence errors

These errors occur while DeltaE's solver is iterating during a `(r)un`:

`This is not going well...DeltaE gives up!` Associated with this error will be a message "info=4", and a listing of the all current guesses followed by all current target−result values. The solver is not able to find an iteration direction that gives improved results. During a plot loop, this error sometimes occurs multiple times, after which the solver once again finds its way. If it persists, a revision to the solution or target vector may be needed, or the starting point (or plot range) may need to be shifted significantly. Examine the `.dat` file for clues, and think carefully about what is occurring. Simplify the model or iteration if possible. If the error occurs on a model that you know to have good convergence under other conditions, you may be reaching a pathological point. You may be able to jump start it by manually (somewhat intelligently) changing the value of one or two members of the guess vector to put the solver on the right track, or, you may find it very stubborn at this point. Consider revising the guess and/or target vectors, or, `(C)lear` all vectors and targets and examine the outputs to see if you can find clues as to the difficulties. Often, the desired targets may not be reachable, given the constraints you have specified. In all cases, if you have spent some effort reaching this point, `(w)rite` the model to save your work because a floating point error that could cause a crash may occur soon.

`Iteration is complete but some results may not be near their targets.` If the text associated with this message is "info=1", and a listing of the all current guesses is followed by all current target−result values, this is a WARNING message, and not strictly an error. It may occur quite frequently. This message is produced by a secondary convergence check that is necessitated by the solver's inclination to be 'satisfied' with agreement that may not meet the users standards. The check is inadequate; it simply asks if the mean square error of targets−results is at least 100 times less than `tolerance` (see Sec IV.H). Based on the relative magnitude of the target values, this threshold may be inappropriate. When this error message occurs during a plot sweep, the line written to the `.plt` file will be preceded by an '∗' to indicate that it requires closer examination. The most common cause of this message is inadequate agreement between results and targets at a `HARDEnd` or `SOFTEnd`. Often,

the message will occur for values that are only slightly off. For a model that has this problem, a good way to judge the quality of the results is to add the residual work or energy flow (parameter `G` or `H`) to the plot list. If the residual flow is orders of magnitude less than the maximum work flow, the accuracy can usually be accepted. You may also consider setting the normalization mode to 2 (see Sec IV.H) to increase the significance of the endpoint errors.

G.2 Input

The following messages can occur when DELTAE is reading in a model description file:

i `numerical parameters expected and only` *j* `were found in` *segtype* `segment, Segment number` *n*. `Edit model file and restart. Last input was:`
This message indicates that an input parameter could not be converted into a floating point value. The value may contain stray, inappropriate characters, or one or more lines may be missing and DELTAE may be trying to read the fluid name as the numerical parameter it needs. For a freetarget, be sure you have specified the initial `Target` value first, even if you do not intend to use it.

`Unknown segment type:` *segtype*. The string at the beginning of the segment description does not match any segments in the library. Be sure that at least the first five characters are UPPERcase. The error could also be stray lines; for example, specifying the plate type twice.

`Illegal fluid:` *fluid string*. This message occurs when DELTAE cannot find the requested fluid in the internal library or as a `fluid.tpf` file in the current directory. Check the spelling of the fluid and be sure that there are enough spaces to fill a 10-character field before any other text occurs. If you are using an external fluid, be sure the file is present in the same directory. If all this appears correct, you may have one line too many of numerical parameters (the giveaway here will be the contents of *fluid string*).

`Unknown plate material:` *plate string*. The comments regarding the `Illegal fluid` message, above, also apply to the plate (solid) specification; however, the default `ideal` solid type may also be specified by a blank line. If this is the intent, be sure that a blank line truly separates each segment module.

`Error reading segment/parameter address in` *segtype* `segment, Segment number` *n*.
This error occurs while reading in one of the freetargets (see Sec. V.B.7) or when processing a `sameas` reference. The characters read do not decode to a valid address in the model.

`More than 5 external fluids found....`

`More than 5 external solids found....`
Only five distinct types of user-defined fluids or plates are allowed in a model at one time.

`Guess/Result vectors are too long. Reduce count before proceeding.` The maximum problem order for this version of DeltaE is 12.

`Too many plot parameters are selected. Reduce count before proceeding.` Up to 13 parameters for plotting can be specified, and the first N of these, where N is the guess vector length, will be selected automatically; these cannot be cleared. You must clear one of the user-selected parameters. In addition, one or two independent variables are also part of the 'plot' file.

`Nested TEE files are not permitted...compile one at a time.` An input file named in a `TEE` statement in turn contains another `TEE` statement. This is not supported. Running DeltaE on the input file first will generate a combined file that can then be included as a branch.

G.3 Model editing

The messages below occur when a model is being modified online:

`*** sameas relationship cancelled...` The parameter you are affecting, by using it in the guess vector, making it an independent plot variable, or `(m)odify`ing it, is not specified directly, but through a `sameas` statement. This connection is severed, and the parameter takes on the value it currently has, until you (or DeltaE) give it another.

`*** Special mode affecting this value must be disabled first.` This parameter is linked back to another parameter that may change, and thereby, modify this value. Such a link is not appropriate if you are trying to set the value independently, or if DeltaE will try to do so while it is plotting or iterating; therefore, you will get the message above when you are try to modify it or make it a guess or an independent plot variable. `(d)isplay` this segment to find the root of this link that must be cleared. It is indicated in () to the right of the parameter description.

`This variable must be cleared from the guess vector first.`
A guess vector member cannot be the target of a link (it *may* be the root), or an independent plot variable, nor may contain a `sameas` statement.

`This parameter is part of a plot loop. It cannot participate in the guess vector.` Using this parameter as a guess would alter the independent variable of the plot loop as the solver iterates.

`This output is not in any vector.` An attempt was made to `(c)lear` a parameter that has not been `(u)sed` or `(p)lot`ted in the target or plot vectors.

`WARNING: could not find appropriate default targets. Modify iteration vectors before solving this model.` While trying to generate a set of guess and target vectors, DeltaE could not find anything suitable. Be sure all `HXFRST, STK*`s, and `HXLAST`s (or `HXMIDl`s) are in proper sequence, and if this is not the problem, DeltaE is not smart enough to help in this case: set your vectors manually.

G.4 Consistency checks

These errors are detected when DeltaE begins processing during a `(r)un`:

`FATAL Error: First segment must be BEGIN.` A `BEGIN` segment is required as the first segment for any model you intend to `(r)un`; without it, DeltaE has no values for the initial conditions.

`SAMEAS parameter types do not match SAMEAS error: Seg#` n, `Parameter` p. Except for freetargets (see Sec. V.B.7), all parameters specified by `sameas` must have a parameter description that matches the root values description through the first four characters. Hence, parameter `a` in a duct may come from `areaI` or `areaF` of a cone, but not from its `length`.

`Circular reference found processing SAMEAS Circular SAMEAS: Seg#` n. This parameter is not rooted in an actual value. It is specified by a `sameas` that, either directly or through additional references, refers back to the same address.

`WARNING: you have` i `guess vector members and` j `target vector members defined. You must either add` k `new target parameters or delete` k `guesses.`
The guess and target vectors have different lengths. You must take some action to balance them.

`Adjustable length segment cannot refer to itself.` The length of this segment (parameter `c`) is either linked to itself or to another segment's length that, either directly or through additional segments, is linked back to this segment.

H Known Bugs and Limitations

- DeltaE's internal solver is very efficient at converging to solutions for complicated systems; however, it knows nothing about acoustics, or any part of physics, for that matter. If it ventures too far, it can give you more wavelengths than you intended before reaching resonance. DeltaE does not know that negative frequencies, negative pressures, or negative lengths are improper; it simply does the math. In short, the reasonableness of the answers produced will almost always depend on the quality of the initial guesses.
- Numbers stored in `.out` files are stored with much less precision than DeltaE actually maintains internally. Sometimes, after storing a file in a particularly difficult computation, the solver will converge a little differently if the file is loaded back in and run again.
- Not all floating point errors are successfully trapped on all systems; some can cause the program to crash and lose unsaved work. Save your work frequently if you are exploring uncharted territory!

I Registration

While there is no formal registration for this program, no fees, and no support or warranty of any kind (please read the copyright notice), we are interested in maintaining a list of users so that we can log any bugs that are found and notify known users of the remedies. If you use this program, please send your name, address, and any comments to Bill Ward, by letter, fax, or electronic mail, at the addresses below. If you find any bugs to report, we would be especially appreciative:


```
Bill Ward
Los Alamos National Laboratory
Group ESA-EPE
MS J576
Los Alamos, NM 87545

Fax:    505-667-0600
E-mail:  ww@lanl.gov
```


News of your successes using this code will encourage our sponsors to consider this effort worthwhile and will enable us to respond to user's questions. Please tell us how this

code has been helpful to you. We are grateful for your acknowledgments in publications and reports and for mention of this work to individuals at agencies that support acoustics research. This will improve our chance to create and pass on improvements in the future.

J Obtaining DeltaE

Fully tested software and user's guide available from Energy Science and Technology Software Center, US Department of Energy, Oak Ridge, Tennessee. We encourage active users to obtain the latest development version directly from us. For a beta-test version available to interested potential collaborators, contact ww@lanl.gov (Bill Ward) via Internet. Up-to-date information on current DeltaE versions is maintained on the World Wide Web server "`http://rott.esa.lanl.gov`".

K Acknowledgments

The development of DeltaE has been supported in part by many agencies and entities: Tektronix Corporation, SPAWAR, the Naval Postgraduate School, and, most importantly, by several branches of the Department of Energy: Advanced Industrial Concepts, Materials Science in Basic Energy Sciences, and our local Industrial Partnership Office, the Technology Transfer Initiative Office, and the Office of Fossil Energy. A long discussion with Pat Arnott helped us define the initial scope of this work, and comparisons with the results of parallel-plate-stack codes (written by Al Migliori and Dick Martin) were useful in the early stages. Suggestions by Kim Godshalk, Charles Jin, Tom Hofler, and Jeff Olson have lead to significant improvements in DeltaE's capability and usability. Charles Jin, Ray Radebaugh, and the code `REGEN3.1` were indispensable in development of the stacked-screen algorithm.